1989年12月1日発行(毎月1回1日発行)第8巻12号通巻92号　昭和58年11月2日第三種郵便物認可

ISSN 0910-7614

パーソナルコンピュータ・マガジン
MZシリーズ,X1/turbo,X68000&ポケコン

Oh!X

オー/エックス 定価560円

特集
Cプログラミングへの招待

付録・C言語簡易リファレンス

Oh!X2周年特別企画
X68000にガイガーカウンタをつなぐ
愛読者特大モニタプレゼント

X1/turbo
アクションゲームACTIVE UNIT

X68000
X-BASIC調理実習/DōGA・CGA講座
マシン語プログラミング

S-OS
SLANG用ファイルリダイレクションライブラリ

THE SOFTOUCH
38万キロの虚空/た～みのる2

LIVE in '89
X1/turbo天空の城ラピュタよりパズーとシータ
X68000ギャラクシーフォースよりBeyond the Galaxy

猫とコンピュータ/知能機械概論
マシン語カクテル/ショートプロぱーてぃ

12
DEC.1989

SHARP

EXPERTシリーズ　本体＋キーボード＋マウス・トラックボール
CZ-602C-BK（ブラック）・-GY（グレー）　標準価格356,000円（税別）
HDタイプCZ-612C-BK（ブラック）　標準価格466,000円（税別）

PROシリーズ　本体＋キーボード＋マウス
CZ-652C-GY（グレー）・-BK（ブラック）　標準価格298,000円（税別）
HDタイプ　CZ-662C-GY（グレー）・-BK（ブラック）　標準価格408,000円（税別）

選べる3タイプのディスプレイをサポート

15型カラーディスプレイテレビ（ドットピッチ0.39mm）　CZ-602D-GY（グレー）・-BK（ブラック）　標準価格 99,800円（チルトスタンド同梱・税別）
15型カラーディスプレイテレビ（ドットピッチ0.31mm）　CZ-612D-GY（グレー）・-BK（ブラック）　標準価格119,800円（チルトスタンド同梱・税別）
14型カラーディスプレイ　（ドットピッチ0.31mm）　CZ-603D-GY（グレー）・-BK（ブラック）　標準価格 84,800円（チルトスタンド同梱・税別）

資料請求券
X68000
Oh!X
12係

表紙絵：Moto Noriyuki

■広告目次

Oh!X

CONT

〈スタッフ〉
●編集長／前田 徹 ●編集／植木章夫 太田慎一 岡崎栄子 ●協力／有田隆也 中森 章 後藤貴行 林 一樹 荻窪 圭 岡本浩一郎 毛内俊行 吉田賢司 影山裕昭 相馬英智 古村 聡 村田敏幸 丹 明彦 三沢和彦 長沢淳博 宮島 靖 金子俊一 ●カメラ／杉山和美 ●イラスト／永沢しげる 山田晴久 小栗由香 ●アートディレクター／島村勝頼 ●レイアウト／元木昌子 AD GREEN ●校正／千野延明 織田洋子

1989 DEC. 12

E N T S

X68000にガイガーカウンタをつなぐ

ACTIVE UNIT

TETROCK

38万キロの虚空

サイクロンCG大会

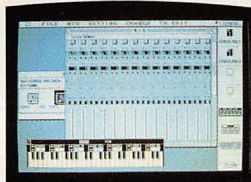
IT X640/680

X68000をサポート。

シャープオリジナルソフトウェア

X68000

サウンドツール

Musicstudio PRO-68K ver.1.1
■CZ-252MS 標準価格28,800円(税別)
24トラック対応MIDIマルチレコーディングソフトMusicstudio PRO-68Kがバージョンアップしました。従来の機能に加え、小節間のコピー及びデリートや、MIDIインプットモニターなど、数々の機能を追加・改良。さらに使いやすくなりました。
※MIDIボード(CZ-6BM1)が必要です。

MUSIC PRO-68K〔MIDI〕
■CZ-247MS 標準価格28,800円(税別)
MIDI対応自動伴奏機能をサポート、簡単な楽譜入力で演奏が楽しめます。
※MIDIボード(CZ-6BM1)が必要です。

ソングライブラリ〈101曲集〉
■CZ-248MS 標準価格8,800円(税別)
鑑賞用と音楽データ加工作成用からなるライブラリです。

Sampling PRO-68K
■CZ-215MS 標準価格17,800円(税別)
AD PCM機能を活かす高機能サンプリングエディタ。多彩なEDITORを装備、サンプリング音のデータはBASICでも活用できます。

SOUND PRO-68K
■CZ-214MS 標準価格15,800円(税別)
スタジオのコンソールパネルを操作する感覚でFM音源による音創りが楽しめるサウンドエディタ。

MUSIC PRO-68K
■CZ-213MS 標準価格18,800円(税別)
最大8パートのスコア(総譜)が書け、内蔵のFM音源で演奏できる楽譜ワープロ&演奏用ツール。

アートツール

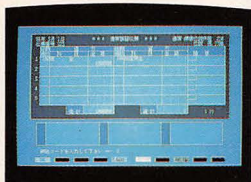
NEW PrintShop PRO-68K
■CZ-221HS 標準価格19,800円(税別)
オリジナリティあふれるはがき等、簡単に作成、印刷できるホームプロダクティビリティツール。ほとんどの処理をアイコンで表示しマウスで選ぶフレンドリーオペレーション。

グラフィックライブラリ VOL.1
■CZ-235GS 標準価格8,800円(税別)
暑中見舞用を中心としたNEW Print Shop PRO-68K用グラフィックデータ集。

グラフィックライブラリ VOL.2
■CZ-236GS 標準価格8,800円(税別)
年賀状を中心としたNEW Print Shop PRO-68K用グラフィックデータ集。

ビジネスツール

TOP給与計算エキスパート
■CZ-228BS 標準価格200,000円(税別)
給与計算から明細発行までを、リアルイメージ入力により自動的に、素早く処理することができます。

TOP財務会計
■CZ-227BS 標準価格200,000円(税別)
会計エキスパートシステムとデータベースを搭載し、機能と操作性を両立させた財務会計ソフト。

CARD PRO-68K
■CZ-226BS 標準価格29,800円(税別)
自由なレイアウト画面で入力できるワープロ機能を装備したカード型リレーショナルデータベース。

CARD PRO-68K用システム手帳リフィル集
■CZ-241BS 標準価格9,800円(税別)

CARD PRO-68K用活用フォーム集
■CZ-242BS 標準価格9,800円(税別)

DATA PRO-68K
■CZ-220BS 標準価格58,000円(税別)
コマンド入力の手間を軽減するヒストリー機能、罫線ドライバー付レポートライター機能、10進31桁の高精度演算。さらにイメージ表示機能を装備したコマンド型リレーショナルデータベースです。

BUSINESS PRO-68K
■CZ-212BS 標準価格68,000円(税別)
スプレッドシート(表計算)、データベース、グラフ作成機能を緊密に一体化させた統合ビジネスツールです。マウス対応のやさしいオペレーション、高度なエディタ機能、豊富な関数群など、初心者からプロまで幅広く使えます。

ゲーム

シューティングゲーム
〈ツインビー〉
■CZ-217AS
標準価格7,800円(税別)
© KONAMI 1988

シューティングゲーム
〈沙羅曼蛇〉
■CZ-218AS
標準価格8,800円(税別)
© KONAMI 1989

ブロックゲーム
〈アルカノイド〉
■CZ-222AS
標準価格7,800円(税別)
© TAITO CORP. 1987

ドライブゲーム
〈フルスロットル〉
■CZ-231AS
標準価格8,800円(税別)
© TAITO CORP. 1988

スポーツゲーム
〈熱血高校ドッジボール部〉
■CZ-232AS
標準価格7,800円(税別)
© TECHNOS JAPAN CORP. 1988

アクションゲーム
〈パックマニア〉
■CZ-233AS
標準価格7,800円(税別)
© NAMCO

アクションゲーム
〈ニュージーランドストーリー〉
■CZ-230AS
標準価格8,800円(税別)
© TAITO CORP. 1989

通信ツール

Communication PRO-68K
■CZ-223CS 標準価格19,800円(税別)
300〜19,200BPSまでの通信速度に対応し、各種データベースの漢字端末やパソコン通信に利用できます。逆スクロール機能、自動実行機能、コンカレント機能も装備。さらに豊富な編集機能をもった高機能通信ソフトです。

開発ツール

OS-9/X68000
■CZ-219SS 標準価格29,800円(税別)
X68000のもつグラフィック環境はもちろん、AD PCM音声、FM音源とグラフィックの同時再生といったマルチメディア機能をサポート。OS-9のもつマルチタスク機能、リアルタイム機能を活かした使い易く機能的なOS環境を提供します。また、これまでのデータ資産も活かせます。 ※OS-9はマイクロウェア社の登録商標です。

Human68k ver2.0
■CZ-244SS 標準価格9,800円(税別)
システムパフォーマンスを高める処理機能を付加したHuman68kの最新バージョンです。マルチタスクに近い処理環境を提供するバックグラウンド処理、ネットワーク処理、ファイルアクセスのスピードアップなど、さらに高い次元へと進化した機能とユーザーインターフェイス。大容量メディアにも対応。

C compiler PRO-68K
■CZ-211LS 標準価格39,800円(税別)
Cコンパイラ、BASIC-Cコンバータ、アセンブラ、リンカ、デバッガ、アーカイバ、コンバータからなるツール。OS上のプログラム開発を効率良くサポートします。XCはC言語の基本的な仕様に準拠し、ANSI仕様も採用、ハードウェアをサポートした豊富なライブラリ(約700種)も用意されています。

THE福袋V2.0
■CZ-224LS 標準価格9,980円(税別)
アセンブラ、リンカ、デバッガ、アーカイバ、X-BASIC V2.00からなる手軽な開発ツールです。

AI-68K(Staff LISP/OPS PRO-68K)
■CZ-234LS 標準価格188,000円(税別)
AI開発用言語とエキスパート構築ツールがセットになったAIプログラム開発ツールです。

本広告に掲載しております商品および役務の価格には消費税は含まれておりませんので、ご購入の際、消費税額をお支払い下さい。

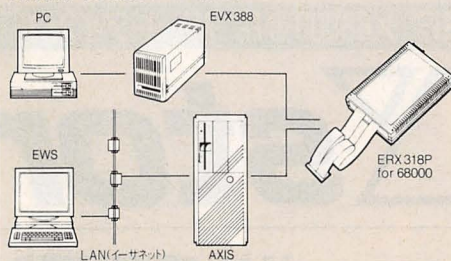
インサーキット・エミュレータ

ICE for X68000

16MHzで快調に動く、ザックスのERX318P for 68000。X68000用ゲームソフト開発に最適です。

クリエイティブワークステーション X68000を、さらにエキサイトさせる開発者へのザックスからの一つの解答。それがインサーキット・エミュレータ、ERX 318P for 68000です。共通ユニットをEVX 388とすると、PC-9801を中心とするMS-DOSマシーンに、またAXISを用いるとEWS（UNIXマシーン）をホストコンピュータとして使用することができます。CGやコンピュータ・ミュージックなど、ゲームソフトの開発には最適なICEです。

ERX 318P for 68000 システム構成図

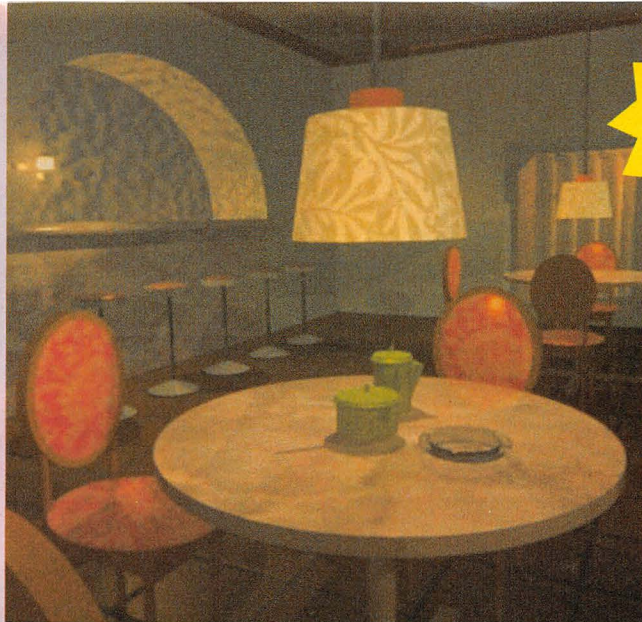
●システム価格：EVX 388＋ERX 68000　￥1,624,000
（消費税別）　AXIS＋ERX 68000　￥2,224,000

ERX 318P for 68000の特長

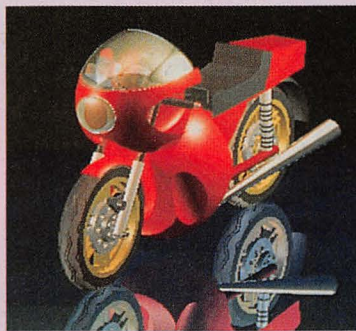
- ●オブジェクトのロード時間：64Kバイトを27秒でロード
- ●ブレーク数：64K×4ポイント（ワイルドカード使用可能）
- ●ホストコンピュータ：PC-9801を中心とするMS-DOSマシン（EVX 388システム）、EWSを中心としたLAN環境（AXISシステム）

Zax Corporation

株式会社 ザックス

本　　社／〒167 東京都杉並区荻窪5-20-12
TEL.03(392)3331㈹ FAX.03(393)3878
大阪営業所／〒532 大阪市淀川区木川東3-5-21 第3丸善ビル
TEL.06(303)2671㈹ FAX.06(303)2454

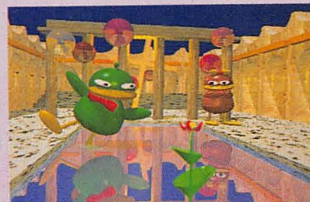
お問合わせはフリーダイヤル 0120-378-388（ICD EVX）

すごい こんなに使える サイクロン

グランプリ賞「SALOON」益津　享　29才

モデリング賞「BIKE」橋元弘司　36才

レンダリング賞「ダンス」田渕友章　29才

SHARP賞「ワキウリ」富田保男

SHARP賞「TAMANORI」駒切　正

LOGIN賞「CHILD」塚田哲也

LOGIN賞「レイトレックに負けるなよ」江畑　一

Oh! X賞　菊池　彰「propose under the moon beam」

サイクロンCG大会を終えて…

使用するハード及びソフトウエアを統一したサイクロンCG大会はとても興味深かった。初の大会でしかも募集期間が短かったにもかかわらず応募された作品はどれも力作ぞろい。対象としたソフトがレイトレーシングということを考えればなおさらである。出品者のソフト使用歴も凄い。なにしろグランプリを受賞した益津氏がわずか2ヵ月であり、応募者の大半が半年未満なのだ。

サイクロンが発売された時期から推測すると、ユーザーの大半がレイトレーシングとごく自然なかたちで付き合っている現実が見えるようだ。サイクロンの存在理由の一番のポイントは、この様に広く一般ユーザーをレイトレーシングという特殊な環境にやさしく迎え入れたところにある。

ビジュアリスト 倉嶋　正彦

参加作品

「タイトル」	名前
●SALOON	：益津　亨
●BIKE	：橋元　弘司
●ダンス	：田渕　友章
●ワキウリ	：富田　保男
●TAMANORI	：駒切　正
●CHILD	：塚田　哲也
●レイトレックに負けるなよ	：江畑　一
●propose under the moon beam	：菊池　彰
●フロイド(3つで1作品)	：江畑　一
●スタースピーダ	：山崎　誠
●スペースソルジャー	：山崎　誠
●インナースペース	：尾崎　真也
●Lover in the sky	：菊池　彰
●幻想の銀河系中心(別名：スモールバン)	：渡久山朝賢
●瀬戸内上空の飛行船	：岩狭　源晴
●少年	：中桐　秀起
●AVERAGE PEOPLE	：阿山　修
●ロボットのおなら	：富田　保男
●幻想	：柳沢　学
●惑星	：柳沢　学
●欲望の適応	：原　志津雄
●パーカー	：松尾　康弘
●CHILD	：塚田　哲也
●ROOM	：羽仁　弘志
●クリスタル	：渥美　清隆

'89年末 SPECIAL NEWS

1 今サイクロン68Kか、Express68を買うと?! そのまま使える便利なマッピングデータ集が付いてくる。

●石、木目、コルク等マッピングデータ20数種類！

2 '90春ポリゴンユーティリティ搭載バージョン発売予定

Z'sトリフォニーデジタルクラフト等の3Dデータがサイクロンで使用可能になる！
そこで、今サイクロンExpress68を買うと?! 無料バージョンアップシールが付いてくる。

※その1・その2共年末限りの企画だよ！お早く各有名パソコンショップまで。
※現サイクロンユーザーの方には別途得々NEWS 等お知らせ致しますのでユーザー登録カードまだの方は必ず御返送下さい。でないと損するよ！！

賞金付

● ポリゴンユーティリティ搭載サイクロンネーミング募集！
☆お近くのパソコンショップ備えつけの応募箱に。

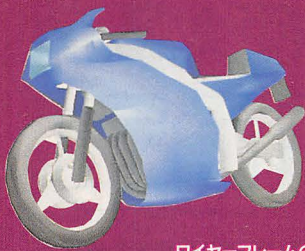
● アンス・キャラクターデザイン募集
☆弊社アンス・コンサルタンツあるいは、サイクロンにふさわしいキャラクターを募集致します。サイクロンを使ってどしどし御応募下さい。もちろん賞金付！！

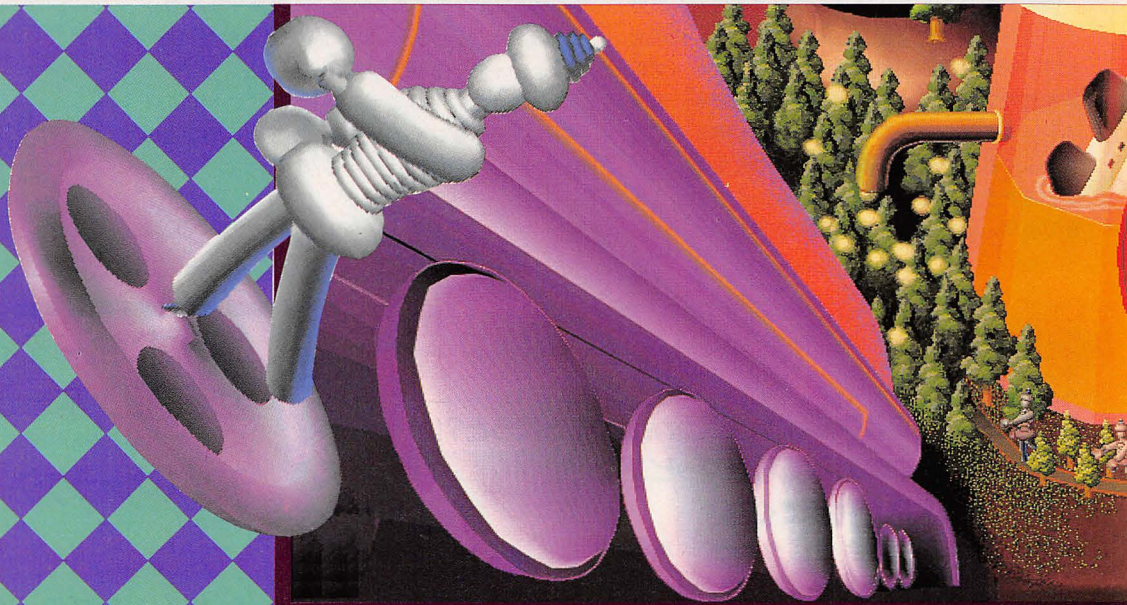
※くわしくは、お近くのパソコンショップまで。

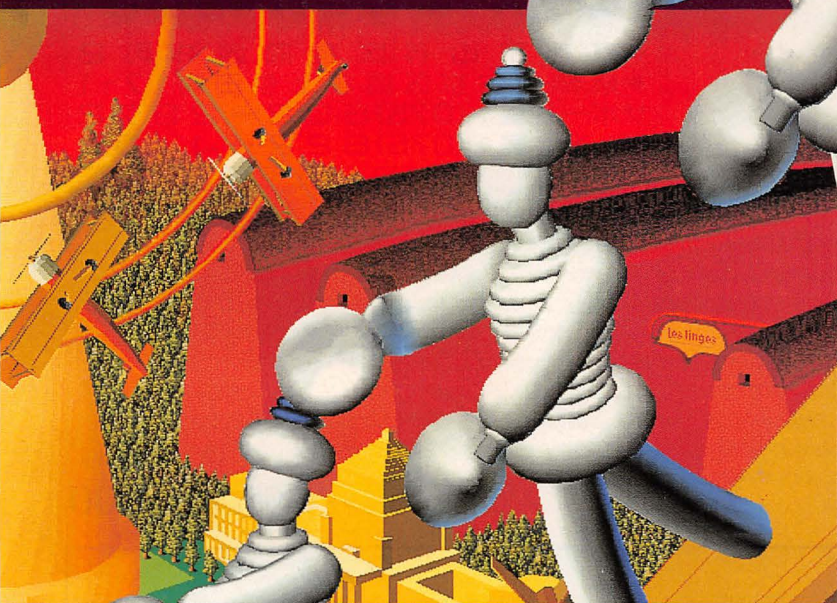
サイクロンExpressシリーズ プロニーズに耐える業務用汎用CGツール

●サイクロンExpress68 ……………… 78,000円
★アプリケーション(別売)……サイクロンアニメキット68Ex……………7,800円

サイクロンVer 1.2シリーズ 3Dレイトレーシング本格入門用CGツール

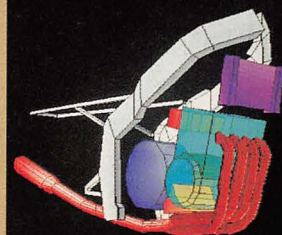
■サイクロン68K ………………………（入門版）58,000円
★アプリケーション(別売)……サイクロンアニメキット68……………5,000円

サイクロンはアンスのオリジナルCG商品です

株式会社アンス・コンサルタンツ
九州本社/〒810 福岡市中央区平丘町68　TEL092-522-6347　FAX092-521-0400
東京本部/〒108 東京都港区高輪2-15-15-203　TEL03-477-4144　FAX03-473-6727

回転体、円柱、面貫通処理などを利用し、イメージを展開してみました。

BY DIGITAL CRAFT

2Dを携えれば、 自己表現の快感は、イ 画面の外にまで ン 揚々とひろがっている。グ。

X68000のハイ・ポテンシャル、65,536色モードに対応したサーフェイスモデルが、マウスだけで緻密に描ける3Dグラフィックソフト、ジーズ・トリフォニィ[デジタルクラフト]。スムース/フラット/ランダム、3種のシェーディングが目的に応じて使いこなせ、高度な面貫通や透明度機能、アンチエリアシング処理を実現。しかも、作成したモデルを最大7cut/secの視点移動により動かせるアニメーション機能を搭載。さらに32,768とすぐれたポリゴン数(3M増設時)を実現しました。プロ・バージョンの2Dグラフィックソフト、ジーズスタッフPRO-68Kとのツインユースで、新しいグラフィックスの悦楽にひたってください。

複雑な造形もオブジェクト毎に作り、それらを組み合せることで容易に、しかも正確に作成できます。

BY DIGITAL CRAFT

エンジン

BINATION

Z'sSTAFF PRO-68K [Ver.2.0]

2Dのプロスペックソフト ジーズスタッフPRO-68K[Ver.2.0] ¥58,000税別

お待たせ致しましたバージョンアップ開始！

NEWS!

OH! BUSINESS
●京都市山科区音羽西林町2
●京都市右京区西院上今田町17-1
サポート室：(075)502－2972
開　発　室：(075)822－4408

エキサイティング・グラフィックツール　G68K VersionII-PRO

G68K Version II-PRO

発売開始

定価：¥22,000

ご案内

この度、弊社では発売中のG68Kをバージョンアップ致します。つきましては、下記のとうりご案内させていただきます。
旧版G68Kは、お求めやすい価格と簡単操作により、入門用ツールとして多くのX68000ユーザーの皆様方よりご好評をいただいております。
今回のバージョンアップでは旧版の簡単操作を継承しつつ、業界でもトップレベルの処理スピードと前作を遥かに上回る、高機能・多機能・高速処理を実現致しました。
旧版G68Kユーザーの皆様方から頂いた多くのご意見を元に、本格的プロ仕様ツールとして大幅バージョンアップ致しました。
サンプルデータもプロのイラストレーターの手に依るコンピュータイラストを収録。また、専用グラフィックデータ集のシリーズ化も予定しております。

高速・高機能・低価格・1MB標準実装のメモリで完全に動作する本格派グラフィックツール。

■前作を大幅に上回る80種類のパレット
- 自由に編集可能
- 模様のついたパレットも作成可能
- HSV方式による色の合成
  色相(色の種類)・彩度(色の濃さ)・明度(色の明るさ)
- 簡単にお望みの色を作り出すための数々の機能を装備
- マスキング塗料・マスク除去塗料を装備
  微妙な修正に威力を発揮
- 2色の混合
- 画面上より自由に色を取り込むスポイト機能
- パレット保存可能
- 画面上より自由にタイルパターンを取り込むタイルパターン用カッターを装備

■32階調の濃淡をもつブラシ
- 自由に形状を変更できるブラシが24種類
- ユーザーが自由に変更・ディスクに保存可能

■大幅に機能アップされたエアブラシ
- ブラシノズル口径、インク噴出速度・濃度を自由に設定

■32階調の濃淡を持つトーンパターン
- 全てのペイントに有効
- 自由に変更・ディスクに保存可能

■強力な編集機能
- 2倍、4倍、8倍に画面を拡大する拡大エディット機能（ルーペ機能）
- 色を調整するカラーコレクタ
- 任意角度の高速画像回転
- 拡大・縮小
- 左右・上下反転
- 切り取りセーブ&ロード
- 自由領域のコピー・移動
- 標準実装のメモリで全画面が編集可能

■製図用具
- マスキング機能
- ペン描画時の直線
- 指定領域のカラー変更
- 円・楕円・ボックス・直線・自由領域
- これらの内部のペイント
- 単色領域ペイント

■文字入力をサポート
- X68000標準24×24ドットキャラクタの表示

■外部機器のサポート
- 豊富な対応周辺機器

■起動直前の画面を保存しながら起動することも可能

■UNDO機能（取り消し処理）
- ペイント等に失敗してもワンステップ前に戻ることが可能

■市販グラフィックツールとのファイルコンバーターが付属
- Z's STAFF-PRO 68Kとのファイル変換が可能

■ノンプロテクト
- ハードディスクへの転送も可能(自由インストール)

■FileはBASICのGL3形式
- BASICより簡単に読み出し可能

バージョンアップについて ▶登録ユーザーにはバージョンアップ案内を送付致します。

旧版G68Kをお持ちのユーザーの方でバージョンアップをご希望の方は、同封の申し込みハガキにてお申し込みください。
お申し込みは、バージョンアップ専用ハガキに限ります。(コピーは不可。)
バージョンアップに際して、旧版のG68Kの返却は必要ありません。今回のバージョンアップは、ディスク・マニュアルの交換バージョンアップではありません。
新版のパッケージ入製品(G68K VersionII-PRO)定価¥22,000をバージョンアップ価格¥10,000(送料込み)にてお届けいたします。旧版G68Kは入門用ツールとしてそのままお使い下さい。
同封のバージョンアップ申し込みハガキを必ずご使用になってください。シリアルNo.を必ずご記入下さい。
同封のDEMOディスクにてニューバージョンの機能を一部ご紹介させて頂きます。是非、ご覧ください。
また、ハガキにて今回お申し込みのユーザー様にはもれなく弊社オリジナルTシャツを商品発送時にお届け致します。

▶お問い合わせ・お申し込みは上記電話番号までお願い致します。(上記サポート室迄)

Arsys

超凄いモノぶちかませ!

ALL DIRECTIONS 3D SHOOTING

©1989 Arsys Software

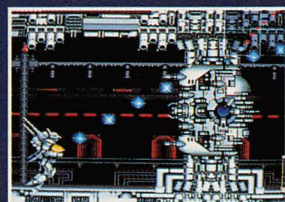
ナイトアームズ

THE HYBLID FRAMER

遥か4億光年彼方の、《かみのけ超銀河団》での戦いには連邦の運命がかかっていた。
宿敵「CIPHER」(サイファー)は、ブラックホールからエネルギーを取り出し、ワープでそのエネルギー球を敵にたたき込むという新兵器「ステラ・スマッシャー」の完成を目前としていた。
対する連邦軍は、「ベース17」基地において、対「CIPHER」用の追撃兵器『ナイトアームズ』を完成させ、前線の「ベース11」へと送り出し、「ステラ・スマッシャー」の破壊を企てていた。
だが、この事をキャッチした敵の先制攻撃に「ベース11」は破壊されてしまった。
残った『ナイトアームズ』はただ1機、「ステラ・スマッシャー」を破壊するために出撃していった。

■縦横無尽に変化する究極3Dスクロール。■鮮やか最高6万色のグラフィック。
■拡大縮少32,767段階!超3Dスプライト。■完全無欠のサイバースティックモロ対応。
■ADPCMのスペシャル効果音。■MIDI真っ青!スーパー音色のウルトラMUSIC。

12月発売予定! WORKS ON X68000 SERIES ¥9,700

©1988/1989 Arsys Software

3D VISUAL

NEW TYPE SPACE ACTION ADVENTURE

スタークルーザー

WORKS ON 好評発売中!

X68000■5″2HD×2 ······ ¥8,800
PC-88SR以降(VA可)■5″2D×2(サウンドボードII・拡張RAM対応) ······ ¥7,800
PC-98M/VM/VX■5″2HD(RAM384k以上、FM音源ボード・拡張RAM対応) ······ ¥7,800
PC-98UV/UX■3.5″2HD(RAM384k以上、FM音源ボード・拡張RAM対応) ······ ¥7,800
X1turbo■5″2D×2(MODEL10不可、FM音源ボード・拡張RAM対応) ······ ¥7,800

開発・発売元:アルシスソフトウェア 〒857 佐世保市松浦町5-13 グリーンビル3F TEL.0956(22)3881
●通信販売のお知らせ●❶使用機種名❷商品名❸住所❹お名前❺電話番号を明記し、現金書留にて弊社へお申込下さい。
※商品の表示価格には消費税は含まれておりません。

■Illustrater: Hitoshi Yoneda ■Sound Producer Keishi Yonao ■Sound Creater: Izuru Aki.

悪夢は終わっていない。

かつて世界は邪教神と悪魔とが人々を震憾させていた。
しかし、この地に住んでいた魔法族・レナムが邪教神を倒し
人々に平和を与えてくれた。

それから1500年。

人々は戦いを忘れ、
レナム一族が消息を絶ったころ
邪教神バーディスが復活しようとしている。
レナムの力と聖剣ソル。

そして…

邪教神を倒す術を求めて、
ソード・マスター：リーク・ラ・ウエルは今、旅立つ。

SWORD OF LEGEND Lenam

レナム

●壮大な世界で繰り広げられるRPGの要素と、奥深いストーリーが楽しめるAVGの要素がドッキング！

●幅広いゲーム展開が楽しめる！

1キャラクター同士の会話等のストーリー展開とモンスターとの戦闘シーンはそれぞれ異なるコマンド・システムを採用。

2ストーリー展開でのキャラクターの移動はワールド・マップが表示。

●今までにない臨場感溢れるBGMが感動させる、先進のMIDI対応音源搭載！

12/15 ON SALE

※画面は×68000のものです。

アドベンチャー・ロールプレイング・ゲーム《レナム》

■X68000 ¥9,800(6枚組)：5″ 2HD/Roland MT32 ※MT32を使用する際はCZ-6BMIが必要です。マウス対応(キーボード入力は不可)

■PC-98 ¥9,800(4枚組)(Vシリーズ以降 要メモリ640KB)：5″、3.5″ 2HD/FM音源対応/Roland MT32/マウス対応(キーボード、ジョイスティックも可)

■MSX2/2+ ¥8,800(4枚組)(VRAM128KB)：3.5″2DD/FM PAC2対応/マウス対応

株式会社ヘルツ
〒169 東京都新宿区北新宿区2-1-16松本ビル3号2F
TEL:03(371)3801 ／ FAX:03(369)4071

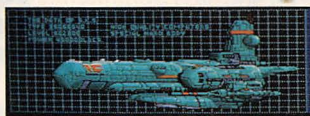
宇宙が、ドラマを語りはじめた。
SLG・ルネッサンス

SCENARIO SIMULATION GAME
シナリオシミュレーションゲーム

狂嵐の銀河
Schwarzschild
シュヴァルツシルト

星暦３９６０年、シュヴァルツシルト銀河外縁部ジロ星団には大小16のさまざまな国々が林立していた。そして、物語はジロ星団の南西部に位置する“サンクリ星国”から始まる—。

時にＫＧＤ星域に遊学中であったサンクリ星国皇太子は、惑星ウーリィに行幸中の父王の暗殺、そして惑星ウーリィの反乱という相次ぐ凶報に、急ぎ帰国の途についた。そして、慌ただしく即位式を済ました後、反乱鎮圧と父王の仇を報じる事に、新王の威信を賭けることとなるのである…。

好評のシナリオSLG 待望のX68000版で 12月遂に登場!!

思わずうっとり、美しいグラフィック
X68000ならではのグラフィックで、
シナリオシミュレーションを超リアルに体験。

好評のシナリオシミュレーション
ＳＬＧなのにシナリオ重視。言葉では言いあらわせないシナリオシミュレーションの魅力をぜひ実際に遊んで味わってほしい。

リアルな戦闘時のアニメーション!
いままでのＳＬＧの常識をやぶった戦闘アニメーションが迫力。

個性豊かな各国のキャラクター達
マニュアルに紹介されている各国の性格が実際にゲーム展開に関わってくるのがシナリオシミュレーションの魅力だ。

■X68000 5″2HD(2枚組)標準価格12,800円12月発売予定
※価格には消費税は含まれておりません。

ねこさんちーむ

●通信販売（送料無料）のお知らせ●
工画堂スタジオでは通信販売をしております。ご希望の方は、品名・機種名・住所・氏名・電話番号を明記の上、３%の消費税を加算して現金書留でお申し込み下さい。

〒162 東京都新宿区市谷台町11 TEL.03-353-7724

激！サイティング！

商品代金
2万円以上
送料無料!!

通信販売でのお申し込みは‥

受注専用フリーダイヤル **0120-377-999** (担当 いせき 井蹟・長倉)
(AM10:15〜PM7:00 休日／12/7(木))
商品についてのお問い合せは各店又は☎(03)251-9911へ

恒例！

秋葉原電気まつり 11/24(金)〜'90 1/7(日)

今年も豪華に賞金総額7000万円でお客様にご奉仕！
5000円以上のお買物でもれなく抽選券をプレゼント！

X68000 PERSONAL WORKSTATION EXPERT / EXPERT HD

特価販売中

CZ-602C
縦置タイプ2M RAM標準搭載
標準価格￥356,000

CZ-612C
40MBハードディスク内蔵タイプ
標準価格￥466,000

X68000 PERSONAL WORKSTATION PRO / PRO HD

特価販売中

CZ-652C
横置タイプ1M RAM標準搭載
標準価格￥298,000

CZ-662C
40MBハードディスク内蔵タイプ
標準価格￥408,000

X68000オリジナルグッズもますます増えて人気上昇中!!
ファンなら集めよう

アクセサリーいろいろ(税別)

- ★ツクモオリジナルキーボード延長ケーブル……ツクモ特価￥1,980
- ★キーボードシリコンカバー……ツクモ特価￥2,400
- ★キーボードセフティカバーASC108……ツクモ特価￥2,400
- ★キーボードダストカバーADC108……ツクモ特価￥1,000

ディスプレイ

型番	内容	価格
CZ-602D	ドットピッチ0.39mmタイプ	定価￥99,800
CZ-612D	ドットピッチ0.31mmタイプ	定価￥119,800
CZ-603D	ドットピッチ0.31mmタイプ	定価￥84,800
CU-21CD	21インチディスプレイ	定価￥139,800

■オプション

型番	内容	価格
CZ-6ST1	(チルト台)	定価￥5,800
CZ-6TU	(RBGシステムチューナー)	定価￥33,100
BF-68PRO	(高性能CRTフィルター)	定価￥19,800

周辺機器

型番	内容	価格
CZ-6BE1	1MB内蔵RAM(CZ-600C専用)	定価￥35,000
CZ-6BE1A	1MB内蔵RAM(ACE・PROシリーズ専用)	定価￥38,000
CZ-6BE2	2MB増設RAMボード	定価￥79,800
CZ-6BE4	4MB増設RAMボード	定価￥138,000
CZ-6BC1	FAXボード	定価￥79,800
CZ-6BP1	数値演算プロセッサボード	定価￥79,800
CZ-6BM1	MIDIボード	定価￥26,800
CZ-6BG1	GP-IBボード	定価￥59,800
CZ-6BU1	ユニバーサルI/Oボード	定価￥39,800
CZ-6BF1	拡張RS-232Cボード	定価￥49,800
CZ-6VT1	カラーイメージユニット	定価￥69,800
CZ-8NS1	カラーイメージスキャナ	定価￥188,000
CZ-6EB1	拡張I/Oボックス	定価￥88,000
AN-S100	アンプ内蔵スピーカーシステム(2本1組)	定価￥36,600

※大好評インテリジェントコントローラー発売中！
これであなたの部屋はゲームセンター……。

CZ-8NJ2……標準定価￥23,800

♪♬ X68000 MIDIセット ♬♪

品名	価格
CM-32L MIDI音源	定価￥69,000
SX-68M MIDIボード	定価￥19,800
CZ-247MS MUSIC PRO-68(MIDI)	定価￥28,800

ツクモ特価￥99,800
(消費税別途￥2,994)

品名	価格
CM-32L MIDI音源	定価￥69,000
SX-68M MIDIボード	定価￥19,800
CZ-252MS Musicstudio PRO-68K Vl.1	定価￥28,800

ツクモ特価￥99,800
(消費税別途￥2,994)

品名	価格
CM-64 MIDI音源(MT-32+サンプリング音源)	定価￥129,000
SX-68M MIDIボード	定価￥19,800
CZ-252MS Musicstudio PRO-68K Verl.1	定価￥28,800

ツクモ特価￥152,000
消費税別途￥4,560

X68000用ハードディスク

ハードディスクがさらに大容量に!!
大特価販売中!!

※ID番号の切り替えスイッチにより増設ドライブとしても使用可能。

アイテック

IT X-640 40MB、28ms
定価￥158,000 ツクモ特価￥128,000

IT X-680 80MB、(40MB+40MB)、20ms
定価￥198,000 ツクモ特価￥158,000

いよいよX68000とデータをやりとりできます

- Stationery PRO-68K
- 専用通信ケーブル
- PA-8500

今年最後の運だめし！！

暮れもやります！コンピュータルーレットでゴーカな景品を当てよう！

12/1(金)〜12/31(日)迄(東京地区のみ)
キミの欲しかった物が当るチャンス！

※1万円以上お買い上げの方に1回のチャンス！
10万円以上は10万円毎に1回追加！

ツクモデンキで今年最後の運だめしだ！

国内・外で活躍！
ツクモグローバルカード

使って便利、持ってて安心。
ツクモグローバルカードは、ジャックス・VISA、セントラル・MCとの提携カードです。ツクモ各店でのお買物がらくらくできる上に、国内はもとより海外での分割ショッピングもOK！しかも18才以上の方なら学生でもOK！！

お申し込みは**03-251-9898**
又は各店頭で…

学生でも18才以上ならOK！

今月のおすすめセット for X68000

オムロン	MD-2400B	300/1200/2400(MNP4)bps対応
SPS	た〜みのる2	X68000用通信ソフト

ツクモ特価¥44,000 消費税別途¥1,320

時代はもうインテリジェントコントローラー

XE-1AJ
標準価格¥24,514(税込)
対応機種：X68000、MSX、PC-8801シリーズ、PC-9801シリーズ
ツクモ特価¥20,836(税込)

XE-1PRO
標準価格¥9,785(税込)
対応機種：MSX、MSX2、FM77AV、MZ-2500、PC-6001、PC-8801 mkIIFR/MR、PC-88VA、X1、X68000(アタリ仕様ジョイスティックボード機)
ツクモ特価¥8,317(税込)

★シャープ製CZ-8NJ2 サイバースティック……………………定価¥23,000

インテリジェントコントローラ対応ゲーム
- PC-9801「AIR COMBAT」システムソフト……………¥8,800
- X68000 「アフターバーナー」電波新聞社……………¥9,200
- X68000 「スーパーハングオン」シャープ……………近日発売
- X68000 「サンダーブレード」シャープ……………近日発売

やっぱり欲しい！カラープリンター

シャープ カラー漢字24ドット熱転写プリンター
CZ-8PC3
定価¥65,800
ツクモ特価¥49,800 消費税別途¥1,494

シャープ カラーイメージジェットプリンター
IO-735X
定価¥248,000
特価販売中

シャープ カラー漢字48ドット熱転写プリンター
CZ-8PC4
定価¥99,800 **特価販売中**
(色は黒、又はグレーを指定して下さい)

おすすめソフトウェア (税別)

★超特価はお電話にてお問い合せ下さい。

- Kamikaze(神風) 統合型スプレッドシート……ツクモ特価¥57,800
- Stationery PRO-68K ステーショナリーツール………定価¥14,800
  ※電子手帳との通信ケーブルは別売です。(¥2,500)
- SOUND PRO-68K サウンドエディタ……………定価¥15,800
- MUSIC PRO-68K ミュージックツール……………定価¥18,800
- Sampling PRO-68K AD PCM活用ソフト……………定価¥17,800
- Musicstudio PRO-68K V1.1 MIDIマルチレコーディングソフト……………定価¥28,800
- MUSIC PRO-68K(MIDI) MUSIC PRO-68KのMIDI版……………定価¥28,800
- ソングライブラリ〈101曲集〉 MUSIC PRO-68Kデータ曲集 定価 ¥8,800
- Communication PRO-68K 通信ソフト………定価¥19,800
- た〜みのる 通信ソフト……………ツクモ特価¥10,900
- た〜みのる2「た〜みのる」Ver Up版……………ツクモ特価¥15,200
- DATA PRO-68K リレーショナルデータベース……………定価¥58,000
- CARD PRO-68K カード型データベース……………定価¥29,800
- システム手帳リフィル集 CARD PRO-68K用フォーム集…定価 ¥9,800
- 活用フォーム集 CARD PRO-68K用フォーム集……………定価 ¥9,800
- Z's STAFF PRO-68K Ver.2.0 グラフィックツール……………ツクモ特価¥49,300
- Z's Triphony DIGITAL CRAFT 3次元サーフェイスモデリングツール……………ツクモ特価¥33,800
- New Print Shop PRO-68K 高機能ポップアートツール……………定価¥19,800
- Terazzo SPRITE EDITOR PRO-68K 高性能スプライトエディタ……………ツクモ特価¥16,800
- サイクロン Express 2.0 レイトレーシングソフトウェア……………ツクモ特価¥67,000
- C-TRACE 68 レイトレーシングソフトウェア ツクモ特価¥57,800
- C COMPILER PRO-68K C言語開発セット 定価¥39,800
- Final X68000 マルチファイル・スクリーン・エディタ‥ツクモ特価¥32,300
- AI-68K AIプログラム開発ツール……………定価¥188,000
- REDUCE 数式処理用ソフト………ツクモ特価¥195,000
- OS-9/X68000 X68000用OS-9……………定価¥29,800
- C＆プロフェッショナルパッケージ OS-9 X68000用Cコンパイラセット……………ツクモ特価¥49,300
- mFORTH Compiler FORTHコンパイラセット……………ツクモ特価¥18,800
- Human68K Ver2.0 Human68KのNEWバージョン‥‥定価 ¥9,800

※その他、ゲームソフトも続々発売中ですので、詳しくはお尋ね下さい。

Music studio データ曲集も特価販売中!! 各¥5,800

- SF-001 国本佳宏／知恵ある暮らしの味
- SF-002 佐久間正英／インセクト
- SF-003 本多俊之／ピーセズ・オブ・ワーク
- SF-004 戸田誠司／あの娘のDNA
- SF-005 佐藤允彦／リゾーム症候群
- SF-006 関根安里／スケッチ

X1 turbo Z III セット

- CZ-888C-BK……………………¥169,800
- CZ-860D-BK……………………¥ 92,200

合計定価¥262,000
限定特価¥158,000 消費税別途¥4,740
24回払い(消費税込)：初回¥8,423＋月々¥7,700×23回払

電子手帳＆ポケコンもやっぱりツクモだよ！(税別)

★いよいよX68000とデータをやりとりできます！

シャープ **PA-8500** 定価¥28,000 **特価¥22,800**
PA-7500 定価¥22,000 **特価¥19,800**

大型4行表示、データスケジュール管理に便利、ICカード、プリンタで更に発展するハイグレードタイプ

シャープ **PC-E500** 定価¥28,800 **特価¥24,800**

シャープ **PC-E200** 定価¥22,000 **特価¥17,800**

モデム (税別)

- オムロン MD12FS(300 1200ボー)‥‥ツクモ特価¥17,800
- アイワ PV-A1200MK3(300 1200ボー)……………ツクモ特価¥14,800
- アイワ PV-A24MNP5(300 1200 2400ボー)MNP5……………ツクモ特価¥45,800

秋葉原各店
至お茶の水 / 不忍通り / 7号店 / 5号店 / カード下奥左側 ニューセンター店 / 三菱銀行 / 万世警察 / シントク / 中央通り / ヤマギワ / JR秋葉原駅 / 山手・京浜東北線 / 至東京 / 至上野 / 至千葉
※AM10時〜PM7時 ㊡年内(12/7のみ)

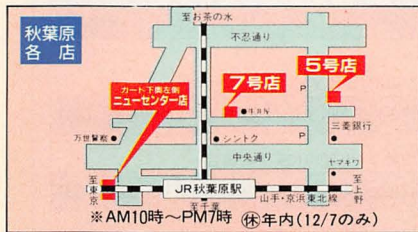
ツクモは「スーパーX PRO SHOP」です。

PRO STAFF ツクモ

九十九電機㈱ 〒101-91 東京都千代田区神田郵便局私書箱135号

ツクモ7号店 ☎03-253-4199

便利で安心な通信販売
通信販売部☎03-251-9911

- ツクモ5号店 ☎03-251-0531
- ニューセンター店 ☎03-251-0987
- 名古屋1号店 ☎052-263-1655
- 名古屋2号店 ☎052-251-3399
- ツクモ札幌 ☎011-241-2299

カード払い	全国代金引き換え配達	クレジット払い	現金書留払い	銀行振込払い
通信販売での御利用カード：ツクモグローバルカード、VIPカード、セントラル、ジャックス 御本人様より電話で通信販売部へお申し込み下さい。	お申し込みは☎03-251-9911へお電話1本！ 配達日の指定もできます。	月々¥3,000以上の均等払いも頭金なし 夏・冬ボーナス2回払いも受付中	〒101-91 東京都千代田区神田郵便局私書箱135号 九十九電機㈱通信販売部Oh！X係	事前に☎でお届け先をご連絡下さい 富士銀行 神田支店(普)No.894047

※11/20(月)より営業時間がAM10:15〜PM7:00に変わります。

★表示価格には消費税は含まれておりません。

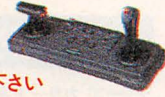
超低金利クレジットOK

またまた秋葉原でおなじみの

注目!!
平成2年1月末一括払いOK!!
手数料(金利)無料!
(平成2年1月末払いをご利用下さい)

11/15〜12/15

●お近くの方はお
●本体単品で特
●ビジネスソフト定

CYBER STICK
●CZ-8NJ2
(定価¥23,800)
超特価!!
▶価格はTEL下さい

X-1ターボZIII 特別ご提供品!! 台数限定
●CZ-888C+CZ-860D+M-2HD(10枚)
定価¥269,600▶特価¥164,800
(ボーナス併用も有りますTEL下さい)
・ジョイカード
・ゲーム3種
・パソコンラックⒶ3段
プレゼント中
送料消費税込み!!

12回	14,300	24回	7,500	36回	5,100	48回	4,000	60回	3,300

ジョイスティック 送料¥500
●X-1PRO
定価¥9,500▶特価¥7,800
●ASCII STICK
定価¥6,800▶特価¥5,500

P&A超低金利クレジットをご利用ください!!

X68000EXPERT & EXPERT-HD (送料消費税込み)

EXPERT&PROセットでお買い上げの方に
●ディスケット(10枚)
●ゲーム
●アフターバーナー(定価¥9,200)
●CZ-8NJI(ジョイカード)
プレゼント中

EXPERT
(ボーナス併用も有ります。TEL下さい)

Ⓐセット:CZ-602C+CZ-603D……定価¥440,800▶現金価格はお電話下さい

12回	29,100	24回	15,200	36回	10,500	48回	8,100	60回	6,800

Ⓑセット:CZ-602C+CZ-602D……定価¥455,800▶特価(現金価格はお電話下さい)

12回	30,700	24回	16,100	36回	11,000	48回	8,600	60回	7,100

Ⓒセット:CZ-602C+CZ-612D……定価¥475,800▶特価(現金価格はお電話下さい)

12回	32,000	24回	16,700	36回	11,500	48回	8,900	60回	7,400

Ⓓセット:CZ-602C+CU-21CD……定価¥495,800▶特価(現金価格はお電話下さい)

12回	32,500	24回	17,000	36回	11,700	48回	9,100	60回	7,600

EXPERT-HD

Ⓐセット:CZ-612C+CZ-603D……定価¥550,800▶特価(現金価格はお電話下さい)

12回	35,900	24回	18,800	36回	12,900	48回	10,000	60回	8,400

Ⓑセット:CZ-612C+CZ-602D……定価¥565,800▶特価(現金価格はお電話下さい)

12回	37,800	24回	19,800	36回	13,600	48回	10,600	60回	8,800

Ⓒセット:CZ-612C+CZ-612D……定価¥585,800▶特価(現金価格はお電話下さい)

12回	38,700	24回	20,300	36回	13,900	48回	10,800	60回	9,000

Ⓓセット:CZ-612C+CU-21CD……定価¥605,800▶特価(現金価格はお電話下さい)

12回	39,300	24回	20,600	36回	14,100	48回	11,000	60回	9,200

X68000PRO & PRO-HD (送料消費税込み)

EXPERT&PROセットでお買い上げの方に
●ディスケット(10枚)
●ゲーム
●アフターバーナー(定価¥9,200)
●CZ-8NJI(ジョイカード)
プレゼント中

PRO
(ボーナス併用も有ります。TEL下さい。)

Ⓐセット:CZ-652C+CZ-603D……定価¥382,800▶特価(現金価格はお電話下さい)

12回	25,200	24回	13,200	36回	9,100	48回	7,000	60回	5,900

Ⓑセット:CZ-652C+CZ-602D……定価¥397,800▶特価(現金価格はお電話下さい)

12回	26,800	24回	14,000	48回	9,600	48回	7,500	60回	6,300

Ⓒセット:CZ-652C+CZ-612D……定価¥417,800▶特価(現金価格はお電話下さい)

12回	28,200	24回	14,700	36回	10,100	48回	7,900	60回	6,600

Ⓓセット:CZ-652C+CU-21CD……定価¥437,800▶特価(現金価格はお電話下さい)

12回	28,500	24回	15,000	36回	10,300	48回	8,000	60回	6,700

PRO-HD

Ⓐセット:CZ-662C+CZ-603D……定価¥492,800▶特価(現金価格はお電話下さい)

12回	32,800	24回	17,200	36回	11,800	48回	9,200	60回	7,600

Ⓑセット:CZ-662C+CZ-602D……定価¥507,800▶特価(現金価格はお電話下さい)

12回	34,200	24回	17,900	36回	12,300	48回	9,600	60回	8,000

Ⓒセット:CZ-662C+CZ-612D……定価¥527,800▶特価(現金価格はお電話下さい)

12回	35,700	24回	18,700	36回	12,800	48回	10,000	60回	8,300

Ⓓセット:CZ-662C+CU-21CD……定価¥547,800▶特価(現金価格はお電話下さい)

12回	36,100	24回	18,900	36回	13,000	48回	10,100	60回	8,400

X68000PRO/ACE-HD〜P&Aスペシャルセット=限定誌上販売!!

30%OFF 送料、消費税別

X-68000PRO 特別ご提供品

台数限定

●CZ-652C(本体)
●CZ-611D(モニター)
●CZ-8PK8(24ピン、漢字、136桁)
+
●ジョイカード(8NJ1)
●ディスケット(10枚)
●ゲームプレゼント中

(定価¥584,000) 特価¥378,000

12回	32,900	24回	17,200	36回	11,800	48回	9,200	60回	7,600

(ボーナス併用も有ります。TEL下さい。)

X-68000ACE-HDセット(台数限定)

●CZ-611C(本体)
●CZ-603D(モニター)
●CZ-8NJ2(CYBER STIC)
+
●ディスケット10枚
●ゲーム
●送料、消費税込み

定価¥508,400 P&A超特価!価格はお電話下さい。

12回	28,700	24回	15,000	36回	10,300	48回	8,000	60回	6,700

モニターをCZ-602D(定価¥99,88)に変更の場合

12回	30,100	24回	15,700	36回	10,800	48回	8,400	60回	7,000

●CZ-612D(定価¥119,800)に変更の場合

12回	31,300	24回	16,400	36回	11,300	48回	8,700	60回	7,300

●CZ-611D(定価¥145,000)に変更の場合

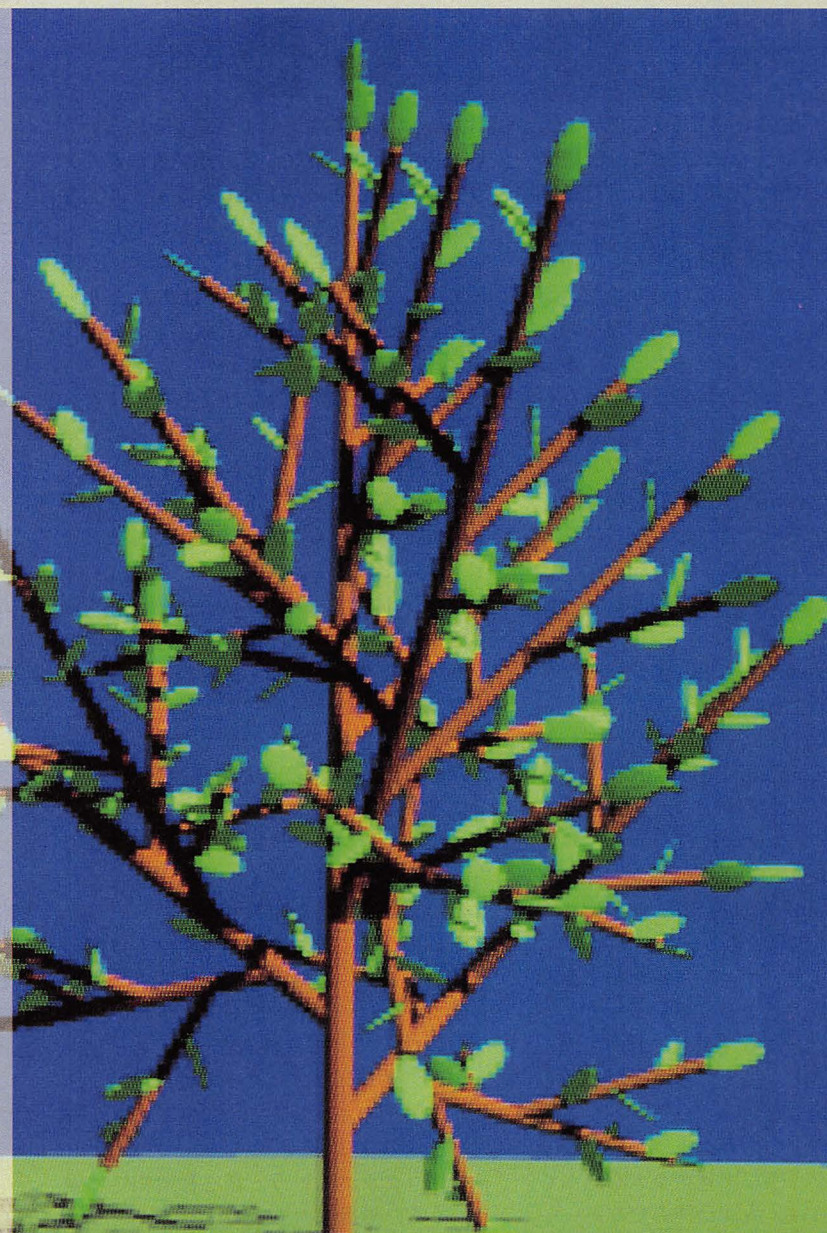
12回	30,700	24回	16,100	36回	11,000	48回	8,600	60回	7,100

(ボーナス併用も有ります。TEL下さい。)

●本広告の掲載の商品の価格については、消費税は含まれておりません。
●営業時間=平日AM10:00〜PM8:00、日祭AM10:00〜PM8:00

特集

Cプログラミングへの招待

「C言語を買ったはいいけれど，いまひとつ使い方がわからない」という方はいませんか？　X68000の場合，XC発売とともに，全ユーザーの2～3割がCコンパイラの正規ユーザーになるという驚異的な普及率を見せました。
しかし本来，C言語はプロ仕様のプログラミング言語です。初めてのプログラミング言語としてC言語を選ぶのは正解ではないかもしれません。一般的な文法自体はALGOL系列に準拠していますので，いわゆる高級言語的な使い方もできますが，そこから1歩踏み出すとたちまち非高級言語的側面が顔を出します。
C言語がC言語たるゆえんは，強力なデータ構造をサポートしながら，余計な変数管理をある程度省略したところにあります。これは本来処理系が考えなければならないことをユーザーが補う必要があるということです。C言語の入門で大切なのは早くC言語とはどんなものかというイメージを確立することでしょう。

C O N T E N T S

扉絵：データ作成　丹　明彦/演算協力　CAST

◉特集　Cプログラミングへの招待

環境設定からコンパイルまで

はじめて使うXC

Ogikubo Kei
荻窪　圭

詳細なマニュアル，ハードをサポートするライブラリ群，加えてBASICコンパイラつき……といえばXCです。ここではXCを使う際の具体的な注意点を順に追ってみましょう。題して，型から入るC言語入門です。

私がこたつを買ったはいいけれど，こたつ布団がない荻窪圭である。世間ではこういうのをただの馬鹿という。早く書き上げてこたつ布団を買わなければ。ぶるぶる。

さてさて，いま手元に業界紙がある。ここに載っているのはソフトハウスのプログラマさんたちはどの言語を使って開発しているか，といった調査結果だが，1位はダントツで“C”。なんと40%以上である。2位は金融関係で根強いCOBOLで25%。あとは軒並み10%以下なのである。伝統のFORTRANもIBMさんのPL/Iも最近は凋落。ほかはアセンブラやらLISPやらPROLOGやらBASICやSmalltalk。そしてパソコンユーザーには馴染みのない4GLだ（第4世代言語の略。簡易言語みたいなもの）。

アンケートの対象次第だと思うが，とにかく，Cは凄い。お金の計算に欠かせないCOBOL(といっても，銀行などでかいところはPL/Iに切り替えたみたいだ）など特殊用途以外は，とにかく，Cらしいのだ。Cはそこまでメジャーな言語なのだ。

ずっしりとした重みにつぶされないように

そこで，X68000のもっともメジャーなCコンパイラの(当たり前だ)，XC，C CompilerPRO-68Kである。あの，紙袋が破れるほど重たいC CompilerPRO-68Kである。電話帳セットのようなC CompilerPRO-68Kである。ディスク2枚にマニュアル5冊のC CompilerPRO-68Kである。これだけ厚くて39,800円とは得した気になるC CompilerPRO-68Kである。はあ，はあ。

しかし，困った。マニュアルがたくさんあるのはいいが，どこから手をつけていいかわからない。とりあえずユーザーズガイドを読まねば，と，読んでみるのだが，どうも，よくわからない。

ああ，40%以上のプログラマがこんな厚いマニュアルを片手にプログラムを組んでいるのか。やっぱ，人間は体力だなあ……と挫折している人も多いのではなかろうか。しかし，である。こんな厚いマニュアルを隅から隅まで読むなどという酔狂はそうそういるわけがない。Cなんてのは，根性と押しの一手というのも捨てがたいが，ときにはははずみとシチュエーションですんなりいってしまったりするのだ。やりたいときがしたいときだ。私は，いま，したい。

コンパイルするとは

Cというのはコンパイラである，という言い方はおかしいな。C言語で書かれたプログラムはCコンパイラでコンパイルしないと動いてくれない，といったほうが正しい。コンパイルするとはいうけれど，BASICでインタプリットするとはあんまりいわないな。それだけ，コンパイルという作業は印象的なものなのだね。

BASICがBASIC.Xだけであるのに対し，XCはC.Xがあるだけ，というわけにはいかない。実のところ，CC.X，CCP.X，CC0.X，CC1.X，CC2.Xと5本もあり，それでも足りない。さらにAS.XとLK.Xを必要とするのだ。

CC2.Xはなくてもコンパイルできるので除外しても（実際，使わない人が多い），6本のプログラムを通す必要がある。いくら6本のプログラムが必要とはいえ，ひとつずつ実行していたのではキリがない。実は，使うときにはCC.Xだけ呼べば，すべてをよきにはからってくれるのだ。このCCがCで書かれたプログラムをうんちゃらかんちゃらして最終的に動くヤツを作るために，5本のプログラムを走らせる。

そして，Cで書かれたプログラムは実行形式へと成長していくのだ。まるで稚魚から成魚にかけて名前の変わる出世魚のようなものだ。プログラム名を“Oh!”としよう。Cで書かれたプログラムには“.C”という拡張子をつけることになっているから，ファイル名は“Oh!.C”となる。こいつをソースプログラムという。第2種情報処理風にいうと，原始プログラムというヤツで，ソースというのは“源”の意味である。

この，“Oh!.C”をコンパイルするわけである。まず，CCP，CC0，CC1のコンパイルトリオが一気に“Oh!.C”をコンパイルする。このコンパイルトリオはCで書かれたソースプログラムをアセンブラのソースプログラムに変換する。

コンパイルトリオは翻訳した結果をアセンブラのソースとして出力する。もちろん，Cのソースプログラムは残しておいて，新しくファイルを作るのである。Human68kではアセンブラのソースプログラムは“.S”という拡張子をつけることになっているので（ソースのSだね），“Oh!.S”というファイルが出来上がる。

この“Oh!.S”はAS.Xという68000のアセンブラでアセンブルされ，機械語のプログラムと化す。アセンブルして得られたプログラムをオブジェクトプログラムという。第2種情報処理風にいうと，目的プログラムだ。一般に，オブジェクトモジュール，あるいは単にオブジェクトという。

生成されたオブジェクトモジュールは，Human68kの命名原則に従って，“.O”という拡張子がついたものとなる。“Oh!.O”だ。オブジェクトのOだ。こいつはもう機械語のプログラムなわけだが，実のところ，そのまま実行するわけにはいかない。最後のステップを必要とするのだ。

これをリンクという。第2種情報処理風にいうと，リンクエディット，訳して連係編集（なんのことやら）だ。リンクにはリンカ，LK.Xを使う。

このとき，Cはライブラリなるものを参照する。ライブラリはまあ，BASICでいう“～.FNC”だと思えばいい。Cではほとんどの機能（入出力からお絵描きまで）を外部関数として扱う。こいつらは“～.H”を覗くとたくさん用意されていることがわかる。どのくらいたくさんかというと，ライブラリマニュアルが800ページ以上になるくらいだ。

関数の数だけ“なんちゃら.O”があったらとてもかなわんわけで，全部で4つにまとめてあるのだ。リンカはこのライブラリと生成されたオブジェクトをリンク(結合)

して，実行形式のモジュールを生成する。これをロードモジュールとか実行形式ファイルとかという。もちろん，実行できる形式といえば，拡張子は“.X”である。

めでたく，“Oh!.X”が完成した。拡張子Xがあるということは，Human68kのコマンドのひとつとして認められたということだ。これでもう，氏素姓を問われることはない立派なソフトウェアである。

より楽しくXCを使うために

XCを使うということは，何本ものプログラムをディスクから読み込んで，そのあいだに3本のプログラムをディスクに作り出すということだ。メモリ上でちょいちょいと実行できてしまうBASICに比べてかなり大事業である。実際にはライブラリを読み込んだり中間ファイルを作ったりもするから，ディスクアクセスは上記だけではすまない。よって，時間がかかる。

一般に，CPUに比べて外部記憶装置は遅いという常識がある。もっと遅いのがプリンタで，いちばん遅いのが人間なのだが，まあ，コンパイル時に後者2つは関係ない。問題になるのは外部記憶装置だ。最低なのがフロッピーディスクというヤツである。次がハードディスク，そして，速いのがRAMだ。計算機も人間も疲労コンパイルしないために，コンパイラはよい環境で走らせてやりたい。プログラム開発するときはなるべくRAMディスクを使う，さもなくばハードディスクを使う，これが原則だ。

では，RAMディスクをどう使うかだ。普通は開発するプログラムのあるディレクトリをカレントにする（ファイル名を指定するのに，いちいちドライブ名やディレクトリ名を書くのは面倒でしょ)。それをRAMディスクにすれば，“.S”も“.O”も“.X”もRAMディスクに作られる。これはおいしい。あとからちゃんと必要なものだけディスクにコピーすれば，停電でもしない限りなんの問題もない。

さて，簡単にRAMディスクを使えばいいといったが，ぜーんぶ転送して使おうと思ったら，RAMディスクを大量に確保しないと足りないだろう。メインメモリが2Mバイト以上ないとできない相談だ。あまりでかいRAMディスクをとると，今度はコンパイルができなくなることもある。しかも，辞書などというでかいファイルを置くスペースはないから，辞書はフロッピーディスクとなり，悲しい日本語環境となる。

が，無理してでもRAMディスクは使うべきだ。いくらか妥協して，自分の許せる範囲でやるのがいいだろう。あまり少ないとあっというまにディスクフルになるから（コンパイラが吐き出すオブジェクトやらでかなりエリアを食う)，どんな小さなプログラム開発でも100Kバイト以上は必要だろう。いざとなれば日本語を使わないことにしてASKなんてはずすか（これだけで100Kバイト以上もメモリを使っている)，ローンを組んででもRAMを増設するかだ。

XCが使うファイルをRAMディスクへ

なにをRAMディスクに組み込むか。Cプログラミングに最低必要なファイルを考えてみる。で，CC.Xほか，ディレクトリ，CC，LIB，INCLUDEに入っているものを片っ端から転送する。“〜.H”はインクルードファイル，“〜.A”はライブラリ本体が詰まっている。AS.X，LK.Xも必要だ。

上記を全部足して，388Kバイトとなる。また，CC.Xが必ずアクセスにいくプログラムがある。CASH.Xだ。CASH.Xは指定したファイルを全部メモリ上に読み込んでから使うという，ファイルアクセスを高速化するためのプログラムで，CC.Xが自動的に使ってくれるものだ。絶対必要。

さらに，プログラムと，そのコンパイル結果などが入るスペースも確保しておかないとまずいので，640Kバイトくらい確保しておけばディスクを使わずにすむ。

ついでにAドライブのシステムをCドライブのRAMディスクに転送するバッチファイルを例として載せておこう(リスト1)。

2つのSETコマンドとTEMPコマンドは忘れないように。SETコマンドはライブラリの在処を指定するものだ。CC.Xはこれを見てライブラリを探すのである。TEMPは中間ファイルを作るときのパスを指定するもので，テンポラリファイルの略だ。Cを使うときに限らず，TEMPは常にRAMディスクを指すようにしておこう。TEMPがどんなに有用かは，次のコマンドをTEMP A:とTEMP C:で比べれば一発だ。つまり，

DIR | MORE

である。

無理なく無駄なくコンパイル

実際にCの簡単なプログラムを作ってコンパイルしてみよう。まず，ソースプログラムを作るのである。ここではED.Xを使う。ソースはRAMディスクに置くから，RAMディスクをカレントドライブにして，

ED Oh!.C

である。とりあえず，さっきの例に習って，ファイル名は“Oh!”というわけだ。

さて，まっさらなスクリーンを前にして，初めてCに挑戦する人は困るわけだな。とりあえずCの言語解説書であるCリファレンスマニュアルを読んでも，どうすればCのプログラムができるのかさっぱりわからない。5冊もマニュアルがあるのに，どれも初心者を村八分にしているのだ。困った。さらに初心者用C入門をつけろ！　とシャープにお願いするのもいいが，マニュアルがこれ以上重くなるのも困る。

しかたがないから，Cの入門書を1冊買うのがよろし。特定のCを対象にしたものでなければ（Turbo-CとかMS-Cなど）大丈夫だろう。犯しがちなのが，いきなりカーニハン&リッチー（K&R）の『プログラミング言語C』を買ってしまうことだ。私も第1版65刷（1986年）を持っているが，読んでもよくわからない。ただ，本棚に『はじめてのC』があるとなんとなく恥ずかしいけれども，K&Rの『プログラミング言語C』だと胸を張れる。まあ，それだけ標準であり，権威主義の人には欠かせないわけだ。第2版では読みやすくなったそうだから，1冊揃えておくのもいいだろう。

さて，BASICと違ってCというヤツはとにもかくにもプログラムとして完結したものを提示しないとコンパイルしてはくれない。しかも，Cはほかの言語と違って，とっつきにくい。普通の言語がよく目指す“英語の文章に近いもの”でさえないのだ。もっとも，日本人にとっては英語に近くてもたいしたメリットはないのだが。

Cでは英単語の代わりに，記号がたくさん出てくるのでどっちにしても困る。たとえば，プログラムのあるまとまり（ブロック）をPascalではbeginとendで囲って示す。PL/Iではdoとendである。それに対してCでは {と} だ。と，こんな調子である。

リスト1　xccp.bat

```
 1: echo off
 2: copy bin¥cc.x c:
 3: copy bin¥as.x c:
 4: copy bin¥lk.x c:
 5: copy bin¥cash.x c:
 6: copy bin¥ed.* c:
 7: copy cc¥cc*.x c:
 8: md c:include
 9: cd c:include
10: copy include¥*.h c:
11: cd c:¥
12: md c:lib
13: cd c:lib
14: copy lib¥*.* c:
15: cd c:¥
16: set lib.=c:¥lib
17: set include=c:¥include
18: temp c:
```

こういったことがたくさんあるから，人の作ったプログラムを見て勉強するのも面倒臭い。

そこで，私としては，“型から入るC”を提案しよう。空手でもお茶でも入門者は型から教えられる。型を真似ていくうちに，最終的にその精神を学んでいくのだ。日本人の得意な戦術である。どうしてそうなるのかはさておいて，疑わず型から入る。最終的に型を破らないと一人前とはいえないので，注意されたい。

では，基本形を見てみよう。以下は，どこにでも転がっているサンプルプログラムである。

```
#include        <stdio.h>
main ()
{
        printf (“ヤッホー¥n”) ;
}
```

と，こんなもんだ。BASICでいうと，

```
10 print “ヤッホー”
```

と，まったく同じである。が，すでに似て非なるものだな。なんといっても行数が違う。しかし，Cにとって行数は意味を持たない。BASICが改行をひとつの単位とするのに対し，Cでは{ }とか，必ず関数（なんちゃら（うんちゃら）という形で完結するものはたいていそうだ）やら代入文やらの後ろにあるセミコロン，といったものがひとつの区切りになるわけで，2行にまたがったり，1行にまとめるために“：”をつけなきゃならない，ということはない。

上の例だと，

```
main () {printf (“ヤッホー ¥n”) ;}
```

でもいいわけだが，まず，そうやって書くことはしない。これが“愛は負けても親切は勝つ（カート・ヴォネガット）”という精神だ。一種の不文律といってもいい。短いプログラムだからいいようなものの，長いヤツでこんなことをされた日にはたまったものではないからね。ま，BASICだと書き方でメモリを節約できたけど，コンパイルすると1行で書こうが何行にも渡って書こうが結果は一緒。なら，綺麗なほうがいいのは当たり前だの編集長だ。

さて，とりあえず，これを入力してセーブし，コンパイルすればいいわけだが，ただ“動いた！”と喜んでみてもしょうがないので，解説を入れるのである。

まず，1行目だ。いきなり#includeで初心者はこう思う。

「なんやようわからんけど，とりあえずつけとけば動くやろ」

それはそのとおりだが，型だけ覚えてもつまらないので，一緒に意味も覚えよう。

#includeというのは，#とincludeに分解できる。#というのは，このコマンドは“プリプロセス”命令ですよ，という印である。いうなれば，Cのコマンドではないのである。Cなどのコンパイラではコンパイルする前にプリプロセッサを通してプリプロセスコマンドをCのプログラムに展開するのである。これは，CCP.Xが行う。

さらに，includeは“（全体の一部として）含める”という意味である。つまり，コンパイルする前に#includeコマンドに書いてあるファイルを取り込んでちょうだい，という命令なのだ。< >の中のファイルがその取り込むファイルであり，includeファイルである。includeはインクルードという外来語的な定着をしているので，覚えておいたほうがいい。

では，インクルードするCのライブラリ“stdio.h”にはなにが入っているのか。ちなみに，stdioというのは“スタンダードI/O”の略，日本語にすると標準入出力だ。知らずに“スタジオ”ってなんだ？　などと大ボケをかましてはいけない。で，ここには標準入出力関係の関数が入っている。

ここらでCの原則を示さねばなるまい。Cという言語は，入力やら表示からグラフィックまで一切の機能を持っていない。つまり，手足目耳口鼻といったものがない百鬼丸だ。すべてのそういった機能は関数という形で提供される。だから，この場合のstdio.hには，関数“printf”など（正確にはその宣言）が入っているのである。

続いて，2行目のmainである。これはmainと書いてあるから，メインルーチンなんだな，と思うのは間違っていない。問題はその後ろの（ ）である。なんで（ ）があるのか？

正解は簡単。mainという名前の関数を定義しているからである。関数は引数を持つことができる。よって，引数がない場合でも（ ）をつける必要があるのだ。Cではメインルーチンでさえ関数なのである。名前はmainと決まっている。で，Cでは関数名を書いたあと，{ }のあいだに書いた命令たちでもってその関数の仕事を表す。mainもその例に洩れない。

では，ED.Xで入力する。気をつけるのは，ふつうのBASICと違って，予約語でも大文字小文字が区別されるということだ。たとえば，includeとか，main は小文字でなければならない。注意ね。

それでもって，字下げだが，普通，TABキーを使う。関数名は1桁目から書いて，その中身はTABキーを押して9桁目からだ（編注：ただし掲載リストはスペースの関係上字下げを小さくしています）。

これでよし。あと注意するところは，printfの中ね。ヤッホーの後ろの“¥n”だ。これは改行しなさいという意味である。

コンパイルするぞ

では，Oh!.Cをコンパイルしよう。

```
cc Oh!.C
```

でいいのである。とりあえずは。すると，しばらく待たされて，図1のような結果になるだろう。リンクではちょっと時間がかかるのでご容赦を。

ここで気になるのがエラーメッセージである。3行目でWarningだ。ワーニング，つまり警告である。関数の戻り値がないんだけれど，大丈夫か？　と，親切なメッセージだ。さて，ここで君ならどうする。ま，何行目でどんなエラーが出たか覚えておいて，もう1回“ED Oh!.C”とやるか？

それでもいいけれど，長いプログラムでコンパイルエラーがたっくさん出たときなんかはいちいち覚えてられないし，書き留めるのも面倒臭い。そこで登場はエディタのタグジャンプ機能というヤツだ。

手順を説明するぞ。マニュアルにもよく使い方が載っていないタグジャンプだ。

1) コンパイルするとき，“/E”オプションを使う。Eは大文字だ。
2) すると，画面のメッセージは簡潔なものとなり，代わりに，“CC.ERR”というファイルが作られる。画面にメッセージが出ないのが，ちょっと，不安。
3) ED CC.ERR で，エディタにCC.ERRを呼び出す。
4) ここで，エラーメッセージの行にカーソルを合わせ，ESC-Vだ！
5) すると自動的にコンパイルしたファイ

図1　表示メッセージ

```
X68k CCP Pre-Processor v1.01 Copyright 1987 SHARP/Hudson
X68k CC0 Parser v1.01 Copyright 1987 SHARP/Hudson
oh!.c    5    :Warning 15 関数の戻り値がない
X68k CC1 Code Generator v1.01 Copyright 1987 SHARP/Hudson
X68k Assembler v1.01 Copyright 1987 SHARP/Hudson
No Fatal error(s)
X68k Linker v1.01 Copyright 1987 SHARP/Hudson
```

ル(この場合はOh!.C)が読み込まれ，エラーのあった行にカーソルが飛ぶのだ。

6)　ESC-Dでファイルが切り替わるので，4)→5)→修正→ESC-Dを繰り返す。

以上。タグジャンプ機能であった。たいていの場合はコンパイルエラーが起きるので，/Eオプションは欠かせない。

エラーが多いときは，下から修正していくのがコツだ。上から順に直していくと，途中で挿入や削除をしたときに行番号と行の対応が狂ってしまうからね。ついでにコンパイルエラーの話をもう少ししよう。

コンパイルエラーには3種類ある。警告のほかに，致命的なエラー，コンパイル時のエラーだ。警告以外のエラーでは，だいたい途中で止まる。エラーがあるのにオブジェクトファイルを作ってもしゃあないというわけだ。が，エラーがなかったからといって，そのプログラムが動くとは限らない。なんの警告もなくコンパイルが完了したプログラムだって，簡単に暴走したりバスエラーですといって止まってしまったりするのだ。エラーの話，終わり。

で，Oh!.Cでタグジャンプをしたりすると(しなくても一目瞭然だけど)，5行目は最後の行だということがわかる。

これはなにかというと，main関数は値を返すようになっているのである（戻り値を宣言しない関数は自動的にintの値を戻すと見なされるのだ)。なのに値を返さないから，警告が出たのだ。これは無視してもよい警告である。この警告が出るのを嫌うときには，最後に，

return (0);

を加えればいい。これは，戻り値（英語でいうと，リターンコード）0を返す命令である。mainからの戻り値はバッチファイルのifコマンドのERRORLEVELオプションで見ることができる。あるいは，mainの前に，voidをつけるのもいい(mainを戻り値のない関数として宣言してしまう)。

無事，Oh!.Xというコマンドができたであろうか。なに？　アセンブル時に出た“No Fatal error (s)”はいいのかって？

いいのである。fatalというのは“致命的な”という形容詞だ。つまり，致命的なエラーはないよということである。致命的でないのはあるかもしれないが，そんなことはやってみないとわからないという，親切で正確なメッセージだ。

で，Oh!.Cがコンパイルされた。ここでディレクトリを見ると，Oh!.Sとか，Oh!.OとかOh!.BAKなどを発見するだろう。ついでに，なぜかOh!.Xのサイズだけが大きいことも発見するはずだ。

では，Oh!とコマンドを打ってみよう。

C>Oh!
ヤッホー!
C>

となれば正解。もし，¥nを忘れたりすると，

C>Oh!
ヤッホー!C>

と，マヌケなことになる。

サンプル第2弾

では，X68000ならではのプログラムをいってみよう。ライブラリマニュアルの132ページである。よっこらしょっと。B_EJECT。なんのことはない。ディスクを吐き出すコマンドを作ろうというわけだね。

まず，レベル0とあるけれど，これは，気にしなくてよい。全部の関数はレベル0から3までのどれかだ。0はROMに入っているIOCSをコールする関数だ。1はHuman68kのDOSコールをする関数だ。2はCの関数だ。3はBASIC関連の関数だ。それぞれ，リンク時のライブラリが違う。順に，IOCSLIB.A，DOSLIB.A，CLIB.A，BASLIB.Aに入っている。

一応マニュアルにはレベル2以外の関数を使うときは，違うレベルの関数はあまり併用しないようにとなっている。でも，そうそういつも不都合なわけではないようなので，いいか。

で，ディスクをイジェクトする関数，B_EJECTはIOCS，つまりROMを呼ぶわけだ。マニュアルには丁寧に書式が書いてある。

書式　#include <iocslib.h>
　　　int B_EJECT (DRIVE);
　　　int DRIVE;

である。これは，IOCSLIBをインクルードしなさい。それでもって，関数の宣言は，戻り値がint（つまり整数）で，引数もintだよん，である。馬鹿正直に，プログラムにこのとおり書く必要はない。だいたい，この宣言はインクルードファイルに入っているのだ。だから，プログラムにはインクルードかintで宣言するかどちらかがあればいい。実際の関数の中身はリンクのときにあればいいのであって，ここでは宣言だけしておけばかまわないのだ。

私としては，#includeより，プログラムの中でどんな関数を使っているか把握する意味からいっても，

int B_EJECT (DRIVE);

を入れたい。なんも宣言してないものはintとして扱うというCの癖があるので，戻り値がintかvoidのとき宣言は省略できるのだが，まあ，変に省略する癖がつくとあとで困るから，ひととおりつけておこう。

で，B_EJECTの使い方だ。引数にpda×256を指定します，とマニュアルにはある。で，pdaとは，0x80～83がハードディスク，0x90～93がフロッピーディスクなのだそうだ。ここで，いきなり私はとまどった。0xってなんだ？

0xというのは，16進を表す記号なのだそうだ。BASICでいう&hと一緒だね。とりあえず，0番のディスクをイジェクトしたいので，0x90だ。それに256を掛ける。ん？なーんだ。256というのは16進で100_Hではないか。紛らわしい。つまり，

B_EJECT (0x9000);

とやれば，0ドライブのイジェクトができるようだ。

戻り値は完了コードだそうだ。完了コードってなんだ？　ま，いいか。やってみればわかる。

で，リスト2である。とりあえず，printfを入れ，完了コードをチェックすることにした。%dというのはdecimalのdだ。バッチファイルの%1のようなもので，カンマの後ろの変数に対応する。

さあ，コンパイル。

CC　ファイル名.C

だ。ん？　リンクまでいって，エラーで止まってしまった。ユーザーズマニュアルを見る。どうも，オプション“/Y”(Yは大文字だぞ）をつけないと，リンカさんはIOCSLIB.Aをリンクしてくれないみたいだ。デフォルトではCLIB.A（つまりレベル2の関数）だけしか相手にしていないのね。

しかたがない。もう1回だ。

CC /Y　ファイル名.O

である。この.Oがミソだ。どうせプログラムに問題はないのだから，リンクだけやってしまえばいい。CCさんは偉いもので，渡すファイルの拡張子に応じて必要なものだけを実行してくれるのだ。

実行してみよう。すると，戻り値はディスクがささっているときに2が，ないときには0が返ってきた。めでたしめでたし。

リスト2　eject.c

```
 1: int    B_EJECT(DRIVE);
 2:
 3: main()
 4:     {
 5:     int    RC;
 6:
 7:     RC=B_EJECT(0x9000);
 8:     printf("RC- %d ¥n",RC);
 9:     return(0);
10:     }
```

もう少し高級にしよう

これではいかにも貧弱なので，パラメータに0を指定すればドライブ0を，1ならドライブ1をイジェクトするようにしよう。プログラムを@.Xとすると（なんで@かというとリターンキーに近かったからだ），

@ 0

でドライブ0から，

@ 1

でドライブ1からディスクが飛び出てジャジャジャジャーンというわけだ。となると，プログラムにパラメータを渡してやらねばならない。これができればHuman68kのコマンドができるという寸法だ。ラッキー。

ここで，main関数の引数を使う。そう，main関数は引数を持つことができるのだ。

main (a, b)

とすると，aにはパラメータの数＋1，bには入力したパラメータの内容が入っているメモリへのポインタが入ることになっている。パラメータは複数のこともあるから，bはもちろん配列だ。おっと，初心者泣かせのアドレスとかポインタが出てきてしまった。しかし，気にすることはない。そんなこと知らなくたって，プログラムは書いたように動くのだ。

普通はaではなくargcと，bではなくargvと書くようだ。なぜか知らないが，きっとargcはアーギュメントカウンタ（ARGumentCounter），argvはアーギュメントバリュー(ARGumentValue)ではないかと思う（Vectorだ，という説もある）。

で，どう使うかというと，

```
main (argc, argv)
int  argc;
char *argv [ ] ;
 {
   プログラム
 }
```

とするのだ。argvは配列だから，[ ] をつける。配列の大きさはargcの値によって変わるから書かなくてよい(!?)。で，argvはポインタだと書いたが，宣言はキャラクタのcharだ。ポイントはargvの前のアスタリスクにある。ポインタargvの指すアドレスから始まるデータはキャラクタの配列（文字列）ですよ，という意味だと思ってよい。で，ポインタがあって，中身を取り出したいときはというと，

＊argv [1]

のようにするのだ。後ろの数字は配列の添え字だ。

これを元にして，プログラムを作ってみたのが@.Cだ。残りページの少ない関係で一気にいろんな機能をつけてしまった。

適当に解説するぞ。main( )の前のvoid hlp ( ) というのは，関数の宣言だ。ヘルプメッセージ表示なわけだな。voidだから“return (0) ;”はいらない。

リスト3 @.c

```
 1: #include <stdio.h>
 2: #include <iocslib.h>
 3:
 4: int     B_EJECT(DRIVE);
 5: void    hlp()
 6: {
 7:    printf("使用法: @ ドライブ番号¥n¥tドライブ番号-->{0,1} ¥n");
 8: }
 9:
10: main(argc,argv)
11: int  argc;
12: char *argv[];
13: {
14:    int   RC;                   /* 戻り値の宣言 */
15:
16:    if (argc == 2 )             /*パラメータが1個だったら */
17:       switch (*argv[1])
18:       {
19:          case '0':
20:             RC=B_EJECT(0x9000); /* drive0をejectしてみる */
21:             if (RC == 0)        /* drive0にdiskがなかった*/
22:             {
23:                printf("ドライブ0にディスケットがありません ¥n");
24:                hlp();
25:             }
26:             break;
27:          case '1':
28:             RC=B_EJECT(0x9100);     /* drive1をejectしてみる */
29:             if (RC == 0)            /* drive1にdiskがなかった*/
30:             {
31:                printf("ドライブ1にディスケットがありません ¥n");
32:                hlp();
33:             }
34:             break;
35:          default:
36:             printf("パラメータが間違っています ¥n");
37:             hlp();
38:       }
39:    else hlp();     /*パラメータが1個じゃないとき*/
40:    return(0);
41: }
```

私の場合，関数はメインの前に全部定義しちゃう。これもその一環だ。宣言だけしておいて，中身の定義はmainのあとでやってもよい（そっちのほうが一般的）。

で，main () ではさっきのargc, argvだ。それは，最初のif文で使っている。“argc == 2”というのはargcが2であったら，という意味で，Cでは＝と＝＝を区別している。ちなみに，BASICで≠を表す< >は，Cでは！＝だ。

で，argc == 2というのは，パラメータがひとつだったらという意味だ。そうだったら，{と}の間の処理を行う。ここでswitch文だ。つまりは，＊argv [1]（パラメータの先頭にある文字）が0だったらドライブ0をイジェクトし，1だったらドライブ1をイジェクトし，そうでなかったら，エラーメッセージとヘルプメッセージ，というわけだ。

で，パラメータが0や1でもディスクがないということが考えられる。するとリターンコードに0が返ってくるので，そういうときもエラーメッセージとヘルプメッセージだ。そんでもって，パラメータの数が1でない（多いとか少ないとか）ときもエラーメッセージとヘルプメッセージだ。

私が思うに，どんなくだらないプログラムでも，このくらいは親切にメッセージが出ないといけない。心ばかりのユーザーフレンドリーというわけだ。

このプログラムを理解したら，次は，パラメータがないときはカレントドライブのディスクを吐き出させるようにして，さらに余力があったら，音楽を鳴らしながら吐き出すようにしてもいい。そうして，楽しいコマンドをたくさん作ろう。

BASICでCをする

皆さんご承知のとおり，X68000において，X-BASICはCにコンバートできる。Cにコンバートできるということは，コンパイルしてコマンドが作れるということである。やり方は，簡単だ。

cc　ファイル名.bas

だけでいい。細かい制限はマニュアルを見ていただくとして，ちょいとしたデキ心で，@.cをBASICに書き直してしまった。ポイントはひとつ。Cでしか使えない関数をコメントにしてしまったのだ。

まず，このej.basをコンバートする。この

ままCにしても使えないのはわかっているので，/Cオプションを使って，BASICをCにコンバートした時点でコンパイラの実行を止めるようにする。

さて，ej.cができた。

ここで，エディタを起動し，次の修正を加えるのである。

1) b_init ( ) ;を取ってしまう。これはBASIC環境へのイニシャライズであるから，なくてもかまわない。
2) switch文のb_argvの前にアスタリスクをつける。
3) B_EJECTの前後のコメントを取る。
4) b_exit(0);をただのexit ( );にしてしまう。こうしないと，BASICを終了するとき画面を全部消してイニシャライズしてからHuman68kに戻ってしまうから，メッセージもなにも残らなくて不便なのだ。

こうして得た結果がリスト5のej.cである。あとはこれを，

cc /Y /W ej.c

とやって実行するだけである。/WはBASICのライブラリを使うときのオプションだ。違うレベルの関数を平気で使っているが，問題なく（今回は）動いた。

ej.xが出来上がったら，めでたしめでたし。こんなぐあいで，パラメータの受け渡しさえBASICでできるのだ。Cのおいしい豊富な関数を使いたいけれど，Cを覚えるのはどうも，という人は，試してみるといいかもしれない。たいてい今回のパターンで可能だろう。

もっとも，Cでしか使えないような関数もあるし，結局Cのソースをいじることにはなるのだけれどね。こういうことを始めると，ej.basとej.cが似ても似つかぬものになったりするから管理には気をつけたほうがいい。

腑に落ちないのは，BASICからコンバートしてコンパイルしたej.xのほうが初めからCで書いた@.xより小さく納まっていることだ。うーん，誰か，教えて。

*　　*　　*

もともとBASICというのは，入門用というより，いまも将来もプログラマなんかにはならないだろう人が，なぜかプログラムを書いてなんらかの処理をしなければならない状況に陥ったとき，あまり苦労せずに修得できるよう作られたものだ（と，私は勝手に思っている）。つまり，これからプログラムをたくさん作って遊びたい人向けではないのだ。BASICはBASICであり，決して入門用ではなく，そこで閉じているものだという気がする。だから，OSやコンピュータの知識があまりなくても触れるようになっているのだ。

Cはそうではない。ガシガシでシュタシュタのパワーユーザー向けといえる。なんでもありで，なんにでも使えるのだ。だからこそ，とっつきにくいし,覚える（というよりコツを掴む）のが大変だ。

ま，プログラムを書くなんてまだまだ非人間的な作業である。それを少しでも軽減しようと高級言語が作られてきた。それは少しでも自然言語により近く（英語だけが自然言語というわけでもあるまいに）と，いろいろと考え出されたわけだ。第4世代言語とかいって，エンドユーザーでも使えるような形を目指してきたのだが，ちょっと気のきいたことをしようと思えば低級言語が必要だということは，みんな知っている。そして，パソコンを好きで触っている連中ってのは，みんな気のきいたことをしたくてたまらないものなのだ。

リスト4　ej.bas

```
 10 int        RC,b_argc
 20 if b_argc = 2 then {
 30    switch (b_argv(1))
 40      case '0':ej0():break
 50      case '1':ej1():break
 60      default:print "パラメータが間違っています":hlp()
 70    endswitch
 80                      } else hlp()
 90 end
100 func hlp()
110       print "使用法: @ ドライブ番号"
120       print "          ドライブ番号-->{0,1} "
130 endfunc
140 func ej0()
150 /*   RC=B_EJECT(0x9000);        */
160      if RC = 0 then
170      {
180           print "ドライブ0にディスケットがありません "
190           hlp()
200      }
210 endfunc
220 func ej1()
230 /*   RC=B_EJECT(0x9100);        */
240      if RC = 0 then
250      {
260           print "ドライブ1にディスケットがありません "
270           hlp()
280      }
290 endfunc
```

リスト5　ej.c

```
 1: #include    "basic0.h"
 2: static   int    RC;
 3: static   int    b_argc;
 4: int    hlp();
 5: int    ej0();
 6: int    ej1();
 7:
 8: /********** program start **********/
 9: void
10: main(b_argc,b_argv)
11: int     b_argc;
12: char    *b_argv[];
13: {
14:    if( b_argc == 2 ){
15:       switch( (*b_argv[1]) ){
16:          case  '0':;ej0();break;
17:          case  '1':;ej1();break;
18:          default:;b_sprint("パラメータが間違っています");b_sprint(STRCRLF);hlp();
19:       }
20:    } else{hlp();}
21:   exit(0);
22: }
23:
24: /****************************/
25: int    hlp()
26: {
27:     b_sprint("使用法: @ ドライブ番号");b_sprint(STRCRLF);
28:     b_sprint("          ドライブ番号-->{0,1} ");b_sprint(STRCRLF);
29: }
30:
31: /****************************/
32: int    ej0()
33: {
34:    RC=B_EJECT(0x9000);
35:       if( RC == 0 )
36:       {
37:          b_sprint("ドライブ0にディスケットがありません ");b_sprint(STRCRLF);
38:          hlp();
39:       }
40:  }
41:
42: /****************************/
43: int    ej1()
44: {
45:    RC=B_EJECT(0x9100);
46:       if( RC == 0 )
47:       {
48:           b_sprint("ドライブ1にディスケットがありません ");b_sprint(STRCRLF);
49:           hlp();
50:       }
51:  }
52:
53: /****************************/
```

◉特集　Cプログラミングへの招待

K&Rも知らない

C言語のひ・み・つ

Iwai Ippei
祝　一平

高速，省メモリ，優れた移植性と甘い香りを漂わせるC言語には入門者が陥りやすい罠がいっぱい。Cの伝導者，祝・マハトマ・一平大師にC言語鉄の掟を語っていただきましょう。というわけで，今月のC調言語講座PRO-68Kはお休みです。

しばらく前からの現象であるが，本屋の理工学書のコーナーでは，Cの解説書と一太郎のマニュアル本が山津波を起こしているのである。なにせ『○○゛○○のC』などというウケを狙ったと思われる名前の本まで出てくる始末なのである。さらには，最近になってCをテーマとした雑誌が2誌も創刊されるという状況にもなっている。余計なおせっかいかもしれんが，Cにそれだけの市場があるんかいなとも思ってしまうのである。

そいでもって，時節の変わり目には，やっぱり勘違いを起こす人が出てくるようで，今年35歳になる技術系サラリーマンの梅沢武夫さん（仮名）のように，「ほほう，それでは最近流行りのCでも勉強してみましょうか」などということを考える人がいたりするのである。

そして，梅沢さん（仮名）は，「やっぱりCを始めるんだったら，この本からかな」ということで，現代コンピュータ文明におけるハムラビ法典とさえいわれている『プログラミング言語C』を手に取ったりするのである。

んが，

そのよーな経過でCに参入してきた人の多くが，やがてつまずき，敗れ去っていくはずなのである。実はそこらへんにこそCの奥深い本質があるのだな。

と，ゆーわけで，いま改めてこの，共立出版創業以来のベストセラーといわれる謎の書物『プログラミング言語C』について考え，これからのコンピュータ社会の行末を論じるつもりなのである。

プログラミング言語C　第2版
B.W.カーニハン／D.M.リッチー 著　石田晴久 訳
共立出版株式会社　定価2,800円（税込み）

チェック1:Cは高級アセンブラである

さて，「Cがわからん」という人によくあるのが，「BASICやFORTRANは，かなりやってきたんですけどねぇ」というパターンである。

ふふふふふ。実はだな，BASICやFORTRANやCOBOLを長いことやってきた人ほど，Cにはつまずきやすいはずなのだよ。なんでかというと，Cとは，高級言語の皮をかぶったアセンブラだからなのだな。おおーっと（←死語）いきなりの結論である。で，このことに気がつかないと，Cをモノにすることはできないのである。

注意深く読めばわかるのだが，『プログラミング言語C』（第1版）のまえがきに「Cは，PDP-11上のUNIXを記述するために設計された」ということが書いてある。この，「OSを記述する」というあたりで，アセンブラの臭いがプ～ンとしてこなければいけないわけだな。で，ここらへんのことをいち早く察した人は，あとになって出てくる「i++」とか，「register変数」とか，「extern宣言」とかからも敏感にアセンブラの臭いをかぎとり，かなりすんなりと受け入れることができるはずなのである。しかし，この事実に気づかないでいると，「いったいCでなにをするのか」「Cを使うとなにができるようになるのか」がわからないのだ。そこが最初の落とし穴なのだ。

チェック2:必要性である

そしてだな，これは一般的にいえることであるが，プログラミング言語の習得のためにかなり重要なのが，

その言語を本当に必要としているか？

なのである。なになに，「私だって速くてオブジェクトが小さくてすむCを必要としてる」だって？

あーまーいーぞー。

甘い甘い甘い。その考えはチクロやステビアのように甘い。単に，より速く，より小さいオブジェクトのプログラムが書けるというだけ，すなわち，量的なメリットだけでは，Cを使う必要性は本当はないのである。

ましてや，「そういえば，最近はCが流行ってるしぃ」などという，ナマヌルイ気持ちのプログラマは立ち入り禁止なのである。

それではどーゆープログラマが“本当にCを必要としている”かというと，まずは，マシンの性能を100%引き出さなければならないプログラマであろう。

これはだな，Cがずば抜けてアセンブラと馴染みがよいということに起因しているのだ。たとえば，Cでは「#asm～#endasm」などで，ソースプログラムの中に直接アセンブラのニーモニックを書くことが許されている（もしもそれができないのであれば，そのCコンパイラは入門用のオモチャである）。

そして，Cではユーザーにポインタが開放されているということも重要である。これにより，最悪の場合は，アセンブラでバキバキと書き倒すことが可能になる。プログラマにとっては，これ以上の安全の保証はないのである。

一般的には，たいていの言語でアセンブラで書いたモジュールとリンクし，適当なパラメータを介して呼び出すことは可能になっているが，実はそれだけではあくまで「可能」なだけであって，「使える」領域にはないのである。

そして忘れてはいけないのが「マクロ」である。これがアセンブラ側からの発想であることはいうまでもないが，一般的な高級言語では無視されているこの機能が，実はものすごく便利なものだということは，Cプログラマの常識なのである。

チェック3:ポインタである

C入門者にとって，もっとも明確な脱落地点として「ポインタ」があるわけだな。この点に関しては，もはや同情の余地はない。つまりだな，ポインタがわからないということは，(HL) や (A0) がわからないということであり，ひいては，コンピュータがわかっていないことなのだ。まあ，少なくとも私はそう解釈している。別にアセンブラをバシバシ使えるようになれとはいわないが，ポインタを理解できないというのは，いくらなんでもトホホだと思うのだが，いかがなもんだろうか。

瓢箪から駒:UNIXからC

では，いったいなにがどーなって，このような「高級アセンブラ」が時代の脚光を浴びるようになったのであろうか。

まず，Cという言語は，よく考えてみるとかなりみょーな言語である。最初にその生い立ちであるが，前述のようにUNIXを記述するために作られたわけである。

しかし，そのときには，「それじゃ，これからUNIXというOSを作りますから，まずはそのOSを記述するための言語を設計しまーす」なんてことはなかったのである。

UNIX(の原型)はすでにあって，それはPDP-7のアセンブラで書かれていたわけだな。で，それをPDP-11に移植する際に，またもやアセンブラを使ったわけである。当然ながらそのUNIX(の始祖)上で動いているツールなどもアセンブラで書かれていたわけだ。ところが「いつまでもこないなアホなことはようせんわ」と思ったリッチーさん (K&Rの“R”の人ですね)が，それを使ってUNIXを書き直そうと考えてBとCのコンパイラを作り始めたのである。

つまりだな，目の前に実際的な要求があり，アセンブラでも，すでにPDP-7/11上で動いていたBでもマズイというので，あくまで道具としてCを作ったわけなのだ。

で，こーゆーのは，東洋でいうところの「泥縄」なのである。主役はあくまでUNIXであって，Cではなかったのである。もっとはっきりいってしまえば，Cは副産物みたいなものだったのである。が，いまではCはUNIXの枠を飛び出して，さまざまな方面で開発言語として脚光を浴びているという次第なのである。

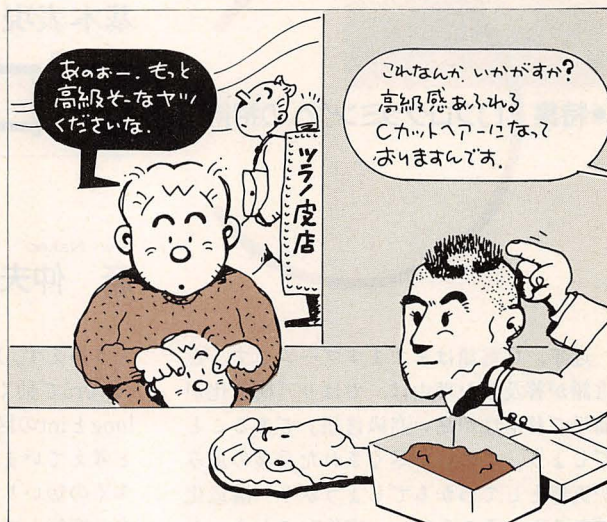
ここから先は私の個人的な考えなので，証明しろといわれるとちょっと困るのであるが，これほどまでにCが使われるようになった理由は，8086の身の毛もよだつようなアーキテクチャが一枚かんでいるのではないかと思っている。まず，事実として，8086/8088のアセンブラでプログラムを書くということは，プログラマにとっては地獄の作業である。

しかし，アメリカでは，IBM-PCがどんどん売れ始めた。IBM-PCは4.??MHzの8088だから，その当時としても決してパワーのあるマシンではない。そこで理想的にはアセンブラなのだが，そうもいかないので，結局，十分に小さくて速いコードを出す，アセンブラの代わりにもなるような言語が必要となる。そしてCがクローズアップされてきたわけだな。

それでも初めの頃はCでゲームを書くなんていうのは，かなり異常なことであった。しかし，試しにやってみると結構できてしまうのである。速度はそれほど遅くないし，拡張性やメンテナスの点ではアセンブラとは比べものにならないほど楽なのだ。このようにしてアプリケーションの開発の主流はCになっていったのであろう。

かくしてCはメジャーになったわけであるが，そのよーな成り上がり言語であるからやることもなかなかに大胆である。まず，高速性とオブジェクトの小ささを優先するので(だってアセンブラなんだもん)，配列の範囲とか，オーバーフローとかは原則としてチェックしない。よって，現存する数少ない「暴走できる言語」のひとつとしてプログラマを楽しませてくれるのである。

これは，BASICやFORTRANやCOBOLやPL/Iに慣れ親しんできた人にとっては「なんでこんないいかげんな言語がもてはやされるんだ？」と頭がこんがらがせるだけであろう。ましてや，なかなか取れないバグと出くわしたなら，うんざりして放り出してしまうかもしれない。しかし，Cを「高級アセンブラ」と見るならば，これはむしろ当然のことなのだな。

で，アセンブラなんだから，当然Cではポインタをユーザーに開放しているわけである。このポインタであるが，はっきりいって，こんな危険なものは，「そもそも存在しない」とか，「使えなくしておく」のが本当は高級なのである。しかしCでは「だってあんたがそうしろってプログラミングしたんでしょ」といいながら，トンでもないメモリ番地に，トンでもないデータを書き込んでしまうのである。

とゆーわけで，Cという言語は大学の教授が設計しそうな言語とはかなり様相が違うわけだ。思い起こせば幾年月。Cが出始めた当時，関心を集めていた言語といえば，PL/IとかModula2とかAdaとか，Prologとか，Smalltalkとかの大見栄を切ったアカデミックな色彩の強い言語であった。ところが，Cはその対極にあり，大勢の人間がCを高級言語と呼ぶことを拒否したほどであった。そのようなこともあったのか，最初の頃はCという言語自体はそれほどは注目されていなかったのである。むしろ注目をされていたのは，(Cで書かれた) UNIXのほうだったといってよいであろう。世の中なにがどうなるかわかったものではない。

ところで，もしOS-9のソースコードが手に入ったとしても，それを新しいCPU (たとえばV70とか80386) に移植するのはかなり大変であろう。その主な理由は，OS-9はアセンブラで書かれているからである (んなわけで，最新のOS-9000はやっぱりCで書かれることになったようである)。

しかし，UNIXは，Cを媒介することによって，以前には想像もできなかった「OSの移植性」を実現してしまったのだ。この事実は，まさにCの「汎用アセンブラ」の一面をよく表しているであろう。

そいでだな，Cの文法はPDPのアーキテクチャの影響を強く受けているということは常識みたいなもんだが，逆に，「Cの文法を反映したCPU」なんてのはどこかにないのかな。つまりCを走らせるために設計されたCPUである。Cとは，そんなCPUが出てくるかもしれないと思わせる言語でもあることよ。ああ，秋深し (しみじみ)。

◉特集　Cプログラミングへの招待

基本表現を覚えよう

プログラミングの定石

プログラムを覚えるときには，他人の作ったものを見て真似していくのもひとつの方法です。ここでは，文字列操作，ファイル入出力など，C言語における基本的なプログラム作成テクニックを具体的に紹介しましょう。

Shin Nakao
新　仲夫

近年，C言語はますますブームです。C言語が普及した理由は，やはり「構造化が簡単で移植性が高い高級言語」であることでしょう。このいい尽くされた言葉の重みが実感としてわかるでしょうか？　構造化できるということは，一度作ってしまった部品はそのまま使い回しができるということです。つまり，プラモデルなどのように，部品を組み立てるだけでプログラムができるのです。また，移植性が高いというのも非常に優れたことでしょう。

とりあえず，プログラミングには「何かを実現する」という目的があります。それは,CGかもしれないし家計簿かもしれません。車の目的が，どこかに移動することであるようなものです。しかし，車自体が好きな人がいるように，コンピュータ自体が目的の人もいるでしょう。そうして，パソコンという限られたハードウェア環境下には,そのような人が多いことも事実です(逆にUNIXにはアルゴリズムが生きがいの人が多い)。しかし，本来の目的はプログラミングではなくその先であるというのは明白です。Cは一応，高級言語ですからハードウェア特有のダーティな部分は気にしなくてもよく(本当は違うのだが)，アルゴリズム作成という純粋に知的な部分のみを考えることができるわけです。

ここでは,なるべく標準のC言語(K&R)に準拠して解説したいと思います。といっても，ANSI (American National Standards Institute) 準拠の第2版ではなく第1版を基に書きます。基本的には，第1版も第2版も変わりはありません。よって，第1版を覚えておけば不自由を感じることはないと思います。また，最近のコンパイラでは関数定義の書式としてANSI提案のものもコンパイルできますが，ここでは伝統的な従来からの書式を採用します(図1)。

リストは，できるだけ基本的なコーディングの約束事のみに従います。つまり，X68000など特定のマシンなどを想定せず，どのマシンでも動くことを目標とします。とりあえず，X68000のXCとPC-9801のMS-Cver5で動くことは保証します。よって，longとintの区別，型変換などは多用したいと考えています。この解説では，なるべく多くの短いリストを載せますので実際に入力，実行してください。また，リストはできるだけコメントを記入しますので本文の解説とともにリストのコメントも参照してください。各リストで使用する関数の仕様は，最後のC言語リファレンスやマニュアルを同時に参照してください。スペースの都合上，リストの実行結果は掲載しませんが，ほとんどが短いプログラムなので，入力してもそんなに時間はかからないでしょう。なお特記しない限りコンパイル・リンクは,

　CC　ファイル名

の形式で行います(XCの環境設定などについては荻窪氏の記事を見てください)。

制御の流れとは

C言語の基本として，まずはprintfから始めましょう。printfは，変数や定数の値を書式に従って画面に出力するものです。Cには定数と変数があります，ここでは，変数に同じ属性の定数を代入して表示するということをやってみましょう（**リスト1.1**)。ところで，Cでは**リスト1.2**のように，文字（char）は整数（int）と同じように扱ってもよいのです。

図1　関数定義のいろいろ

旧）基本表現

```
#include <head.h> ························インクルードファイル
int function(paral, para2)  ···········intを返す関数。引数はparalとpara2
int paral;·······························paralの型はint
char *para2;·····························para2の型はchar*(ポインタ)
```

新）ANSI方式

```
int function(int paral, char *para2)
```

では，制御の流れについて説明しましょう。制御とは，ifとかwhileなどBASICでお馴染みのプログラムを書くうえでの構文です。Cでは以下のものが準備されています。

if-else	分岐処理
while	ループ
for	ループ
do-while	ループ
switch	分岐処理
break	ループ・分岐中止
continue	ループ継続

if-else文は，以下のような書式です。

```
if ( 式 1) {
    式1が真の場合の処理 ;
}
else if ( 式2 ) {
    式2が真の場合の処理 ;
}
else {
   それ以外の場合の処理 ;
}
```

上の例では，{ }(大カッコ)がありますがCでは，宣言や文などをまとめてブロック化することができ，{ }に囲まれた部分をブロックといいます。また，処理の終わりに;(セミコロン)がありますが，Cでは;は文の終わりを意味します。それぞれのifのあとの( )（小カッコ）には，i＝＝0のような任意の式を書くことができ,真(0以外)か偽（0）のいずれかの評価値を持ちます。つまり,整数iが10のとき,i＝＝1やi＝＝2は偽で，i＝＝10は真です。偽のときは常に評

価値は0となります。if文やwhile文などではこの（ ）の中を評価した結果が真だった場合のみ，直後のブロックの処理を行います。

　　if（i）{ 真の場合の処理; }
　　else　{ 偽の場合の処理; }

上の例で，もしｉがϕ以外ならelseは絶対に実行されません。**リスト1.3，1.4**でif文のテストを行ってみましょう。

リスト1.3では最初のifブロックしかメッセージは出ませんが，リスト1.4では全ブロックでメッセージが出ます。つまり，後者ではelseがついていないのでifで判断されたにもかかわらず再度判断の対象になっているのです。後者において，最初のifでiは10なので真となります。次のifでは，まずi－10は0なので偽となり，論理演算子!で否定をとっているので真となっています。( )にはさまざまな式を書くことができます。

while文は，ループ処理を行います。while文は，（ ）の中の式が真であるあいだ処理を実行するものです（**リスト1.5**）。

　　while（式）{
　　　　式が真のあいだ行う処理；
　　}

for文は書式が違うだけで，while文と同じくループ処理を行います。for文は，まず最初に式1を実行して，式2が真のあいだ処理と式3の両方を行うというものです（**リスト1.6**）。

　　for（ 式1; 式2; 式3 ）{
　　　　式2が真のあいだ行う処理；
　　}

リスト1.5と1.6のwhileとfor文は同じことを行ってます。ところで，i－－という表現は，i＝i－1を行うのと同じことです。ほかに，i＝i＋1と同じ処理を行うi＋＋，i＝i＋10などと同じ処理を行うi＋＝10などがあります。リスト1.6は（ ）のあとに｛ ｝がありませんが，１文のみの場合はなくてもいいのです。無限ループは，**リスト1.7**のようになります。

無限ループを抜けるには**break**を使います。また，ループの中での継続処理として**continue**というものがあります。

リスト1.8では，iが真のあいだループします。iは増えるのみなので無限ループです。そこで，iが10になったらbreak文でループを抜けるようにします。また，iが偶数（2で割り切れる数）の場合は以後の処理を行わずループの先頭に戻り，iが奇数の場合はメッセージを出します。while（i＋＋）; とは，while（1）{i＋＋;} と書くのと同じ動作です。ところで＋＋は，それが変数の前につくのとあとにつくのとでは意味が異なってきます。それは評価される順序が異なるからです。ではリスト1.9を見てください。

リスト1.9では，変数aもbも1なのに，表示したら結果が違います。つまり，a＋＋はaの値を評価した（表示した）あと増やしているのに対して，bの場合は増やしたあとに表示しているからです。

switch文は，if文と異なり比較の対象は整数の定数でなければなりません。

　　switch（式）{
　　　　case 整定数1：処理1
　　　　case 整定数2：処理2
　　　　default　　：処理3
　　}

式の値が定数1だったら

リスト1

リスト1-1：変数の表示(test1_1.c)

```
 1: void        main()
 2: {
 3:    /*変数宣言*/
 4:   char  c;                          /*文字*/
 5:   char  str[8];                    /*文字列*/
 6:   int   i;                          /*整数*/
 7:   long  l;                          /*long*/
 8:    /*変数に定数を代入*/
 9:   c='X';                            /*文字*/
10:   str[0]='O'; str[1]='h';           /*文字列*/
11:   str[2]='!'; str[3]='X';           /*文字列*/
12:   str[4]='¥0';                /*最後にNULLを付ける*/
13:   i=100;                            /*整数*/
14:   l=98765L;                         /*long*/
15:    /*表示*/
16:   printf( "char1=[%c]¥n",  c );     /*文字変数*/
17:   printf( "char2=[%c]¥n", 'X' );    /*文字定数*/
18:   printf( "str1 =[%s]¥n",  str);  /*文字列変数*/
19:   printf( "str2 =[%s]¥n", "Oh!X" ); /*文字列定数*/
20:   printf( "int1 =[%d]¥n",  i );     /*整数変数*/
21:   printf( "int2 =[%d]¥n",  100 );   /*整数定数*/
22:   printf( "long1=[%ld]¥n", l );     /*long変数*/
23:   printf( "long2=[%ld]¥n", 98765L );  /*long定数*/
24: }
```

リスト1-2：文字と整数(test1_2.c)

```
 1: void        main()
 2: {
 3:     char  c;
 4:     int   i;
 5:     c=61; /*文字変数に整数*/
 6:     i='A';    /*整数に文字*/
 7:     printf( "c=%d : %c¥n",  c, c );
 8:     printf( "i=%d : %c¥n",  i, i );
 9: }
```

リスト1-3：if文のテスト1(test1_3.c)

```
 1: void        main()
 2: {
 3:     int        i=10;
 4:     if( i ){
 5:        printf( "iは真です¥n" );
 6:     }
 7:     else if( !(i-10)){
 8:        printf( "i-10は偽です¥n" );
 9:     }
10:     else if( i==10 ){
11:         printf( "iは10です¥n" );
12:     }
13: }
```

リスト1-4：if文のテスト2(test1_4.c)

```
 1: void        main()
 2: {
 3:     int        i=10;
 4:     if( i ){
 5:        printf( "iは真です¥n" );
 6:     }
 7:     if( !(i-10)){
 8:        printf( "i-10は偽です¥n" );
 9:     }
10:     if( i==10 ){
11:        printf( "iは10です¥n" );
12:     }
13: }
```

リスト1-5：while文のテスト(test1_5.c)

```
 1: void        main()
 2: {
 3:     int       i=5;
 4:     while(i){      /*iが真(0でない)間*/
 5:        printf( "iは%dです¥n", i );
 6:        i--;     /*iの値を減らしていく*/
 7:     }
 8: }
```

リスト1-6：for文のテスト(test1_6.c)

```
 1: void        main()
 2: {
 3:     int       i;
 4:     for( i=5; i>0; i-- )
 5:        printf( "iは%dです¥n", i );
 6: }
```

リスト1-7：無限ループのテスト(test1_7.c)

```
 1: void        main()
 2: {
 3:     while(1)
 4:        printf( "無限ループです¥n" );
 5: }
```

リスト1-8：break,continueテスト(test1_8.c)

```
 1:
 2: void        main()
 3: {
 4:     int       i=1;
 5:     while( i++ ){
 6:       if( i==10 )break;
 7:       else if( i%2==0 )continue;
 8:       else printf( "%d奇数です¥n", i );
 9:     }
10: }
```

リスト1-9：式の評価のテスト(test1_9.c)

```
 1: void        main()
 2: {
 3:     int       a=1, b=1;
 4:     printf( "a=%d¥n", a++ );
 5:     printf( "b=%d¥n", ++b );
 6: }
```

リスト1-10：switch文のテスト(test1_10.c)

```
 1: #include <stdio.h>
 2: void        main()
 3: {
 4:    int       i;
 5:    while( i!='q' ){
 6:      printf( "キー入力 ->" );
 7:      i=getchar();
 8:      switch(i){
 9:      case 'a': printf("GET IT¥n" );
10:         break;
11:      case 'q': printf("終了¥n" );
12:         break;
13:      default:  printf("入力[%c]¥n",i );
14:         break;
15:      }
16:    }
17: }
```

処理1, 2, 3を実行し，定数2だったら処理2, 3を実行し, まったく該当しなかったら処理3を実行します。

リスト1.10は，画面から入力された文字がaだったらGET ITというメッセージを表示し，qだったら終了，それ以外だったら入力文字を画面に表示するものです。iがaの場合もbreakしているのであとの処理は行いませんが，もしbreakしなかったら以下の文も実行します。getcharは，画面から1文字入力する関数です。#include <stdio.h>は，いまは気にしないでください。

文字列とポインタ

ここでは文字列操作を紹介します。変数にデータをセットして表示する方法はわかりましたが，変数に文字列を代入するのに配列にひとつずつ入れていくのは馬鹿らしいでしょう。よって，文字列単位で操作する方法を紹介します。ところで，文字列とはメモリでどうなっているのでしょうか？ ここで“Oh!X”という4文字の文字列の場合を考えてみましょう（**図2**）。

char str［6］の形式で宣言された文字列は，メモリ上でなんらかの**アドレス**を持っており，str［0］からstr［5］までのアドレスは連続しています（Cでは配列のインデックスは0から）。また，文字列は最後にNULL（'¥0'）が入っており，大きさは実際の文字数＋NULLバイト分が必要です。よって最大10文字の文字列を扱おうとする場合には11バイト分の領域を宣言する必要があります。文字列定数の場合も同じように1バイト余分にメモリに取られます。

ところで，Cでは便利なことにそのアドレスを簡単に知ることができるのです。それは，変数の先頭に&をつければよいのです。つまり，str［0］とはその内容を意味しますが，&str［0］とすればstr［0］のアドレスを意味するのです。さらに，&str［0］と書くのとstrと書くのは同じことなのです。図2の例では，str［0］は'O'でそのアドレス&str［0］は$a1_H$となっています。

さらに，ポインタというものもあります。**ポインタ**は，char　*ptrのような形式で宣言します。ここで注意してほしいのは，変数名は単にptrだということです。ポインタには変数のアドレスを入れておき，実際のデータを見るときは*ptrの形式で指定します。そこで図2の例で，ptr＝&str［0］とポインタにアドレスを代入したら，ptrは$a1_H$で*ptrは'O'となります。よって，char str［6］と宣言した場合，strは文字列を指すポインタであるともいえます。

str［1］をポインタで参照するには，ptrを＋＋して*ptrを見ればいいのです。つまり，配列strのデータをひとつずつ参照するときstr[i＋＋]（文字配列strのインデックスをひとつずつ増やす）という形式と*ptr＋＋として参照するのは同じ結果が得られます。注意してほしいのは，*ptr＋＋とはポインタをひとつずつ移動したあと参照しているのに対して，(*ptr)＋＋は参照しているデータの値を増やす（図2ではstr[0]の値を'O'から'P'に変更するということです。

文字列操作の実際例

さて，ここで第2の文字列str2が定義されたとします。しかし，単にstr＝str2としてはいけません。なぜならば，strは固定されたアドレス（定数）だからです。つまり，str[0]＝str2[0]，str[1]＝str2[1]……と代入していかなければいけないのです。そこで，str1の指す先にstr2の指す先を代入する関数strcpyを使います。ここで注意してほしいのは，str1にはstr2以上の受けエリア（長さ）がないといけないことです。では文字列の長さを知るにはどうしたらいいでしょうか？。それにはstrlenを使います。このように文字列操作のためにはたくさんの関数が用意されています。

ここでは以下の文字列関係の関数を紹介しましょう。

#include <string.h>

文字列s2をs1にコピー（**リスト2.1**）

```
char *strcpy ( s1, s2 )
char *s1, *s2;
```

文字列s2をs1のNULLに連結（**リスト2.2**）

```
char *strcat ( s1, s2 )
char *s1, *s2;
```

文字列sの長さを返す（**リスト2.3**）

```
int     strlen ( s )
char  *s;
```

文字列s2とs1の大きさ比較（**リスト2.4**）

```
int     strcmp ( s1, s2 )
char  *s1, *s2;
```

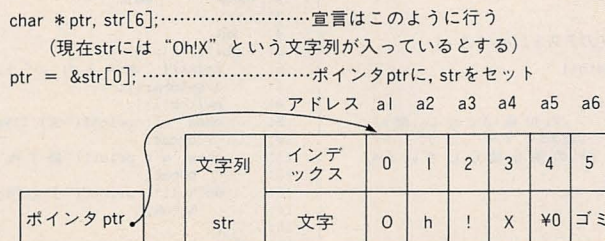
文字列s内の文字cを探す（**リスト2.5**）

```
char *strchr ( s, c )
char *s, c;
```

最後に，簡単な演習をしてみましょう。端末から入力した文字列の前後に入っている空白をスキップして再表示するというプログラムです（**リスト2.6**）。これで，入力コマンドの正しい解釈ができますね。getsは端末から1行分の文字列を入力する関数です。

入出力

入出力（ファイル操作）は，C言語の仕様には入ってない機能です。しかし，そんなことは気にしないでください。私は，入出力操作が，C言語プログラミングでもっとも実用的な項目だと思っています。入出力では，ファイル名とファイルポインタ（XCではストリームポインタ）の2種類の概念があります。ファイル記述子（XCではファイルハンドラ）という概念もありますがここでは触れません。ファイル名とは，A:¥bin¥ed.xとかcommand.xとかいったディスク上の実際のファイル名です。これは，プログラムではあまり意味がなく，一般的にはオープン時のみに使用します。実際の読み書きは，ファイルポインタに対して行います。ディスクにあるファイルを，読み書きするにはOSとさまざまなやりとりをしなければならず，ファイルポインタはその情報を持っているものだと考えてください。ファイルポインタは，ファイルをオープンしたときに得られます。ファイルのオープンとクローズには，fopen/fcloseを使用します。

図2　文字列とポインタ

```
#include <stdio.h>
FILE *fopen ( name, mode )
char *name;
char *mode;
int    fclose ( stream )
FILE *stream;
```

fopenは，ファイル名を示す文字列とモード(mode)を与えます。もし，正常にオープンされればファイルポインタが返され，オープンできなかった場合は，(FILE*)NULLつまり0が返されます。modeは以下のようになっています。

r	読み込みモード。 ファイルが存在しなければエラー。
w	書き込みモード。 ファイルが存在すればファイル長が0になる。存在しなければ新規作成。
a	追加モード。 ファイルが存在すればファイルの終わりから書き込みモードオープン。 存在しなければ新規作成。

ほかに，読み書きモードのr+，w+，a+もあり，データベースなどで同じファイルを読み書きする場合に使用します。

ところで，Human68kやMS-DOSなどではUNIXなどとは異なり，行末の改行文字はCR/LF ($0d_H/0a_H$) の2バイトが使用されています(UNIXなどではLFのみ)。よって，UNIXなどで作ったプログラムは，1バイト余分に読む必要がある場合もあります。そこで明示的にテキストモードかバイナリモードかの指定をすることもできます。指定しない場合は，デフォルトモード(XCでは変数_fmodeに左右)になります。なにも指定しない場合はテキストモードだと思っていいかもしれません。そのモードは，以下のようになっています。

t	テキストモード。 入力時はCR/LFをLFに，出力時はその逆の変換を行う。
b	バイナリモード。 何も変換は行わない。

では，ファイルのオープン/クローズだけのプログラムを見てください(**リスト3.1**)。

ここでは，dataというファイルを読み込みモードでオープンしています。FILEとは構造体の名前ですが，ここでは単にintとかcharのような変数の型だと考えてください。変数名はできるだけ意味のあるものをつけたほうがいいと思います。さっきも出てきましたが，1行目に#include <stdio.h>とあります。実は，これは**プリプロセッサ命令**です。プリプロセッサ命令は先頭に#がつく1文でこれは，絶対に1カラム目から書かなければいけません。プリプロセッサ命令には，#include，#define，#ifdefなど多数があり，非常に強力で便利な機能です。

ファイルのコピー

最初の例題として，test3—3.cからtest3—3.outへファイルコピーするプログラムを紹介しましょう(**リスト3.2**)。ファイルの読み書きには**fgets/fputs**を使います。

fgetsはファイルの内容を1行ずつ変数bufに読み込みます。ここで注意してほしいのは，図のように読み込まれる文字列の最後の1文字はNULL('¥0')であることです。よって，リスト3.2では，一度に最大255文字しか読み込めません。さらに，1行が255文字より短い場合は改行文字('¥n')も入り，長い場合は2度に渡って読み込まれます。

S	T	R	I	N	G	¥n	¥0

ところで，前もって定義されている特別なファイルポインタとして以下のものがあります。また，これらはそれぞれ論理的なファイル名を持っています。なお，stdauxとstdprnは，UNIXにはありません。

ポインタ	名　称	ファイル
stdin	標準入力	con
stdout	標準出力	con
stderr	標準エラー出力	con
stdaux	標準補助入出力	aux
stdprn	標準プリンタ出力	prn

リスト3.2の入出力を標準入出力にしたのが，**リスト3.3**です。Human68kやMS-DOSではconという名前のファイルに読み書きすることもできます。これらのOSでは，標準入力のEOFは^Z(コントロール+Z)の入力で得られます。この^Zについてはあとで解説します。リスト3.3は，以下のように，コマンドラインでのリダイレクション機能を使えば，ファイルに対する入出力も可能になります。

```
test3—3 > outfile
```

リスト2

リスト2-1：strcpyのテスト(test2_1.c)

```
 1: void          main()
 2: {
 3:    char     str1[32], str2[32];
 4:       /*s2に文字列をｺﾋﾟｰ*/
 5:    strcpy( str2, "ここはOh!Xです" );
 6:    printf( "1:[%s]¥n", str2 );
 7:       /*s1にs2をｺﾋﾟｰ*/
 8:    strcpy( str1, str2 );
 9:    printf( "2:[%s]¥n", str1 );
10:       /*s2に&s1[6](Oh以降)をｺﾋﾟｰ*/
11:    strcpy( str2, &str1[6] );
12:    printf( "3:[%s]¥n", str2 );
13: }
```

リスト2-2：strcatのテスト(test2_2.c)

```
 1: void          main()
 2: {
 3:    char     str1[32], str2[32];
 4:      /*初期化*/
 5:    strcpy( str1, "ここはOh!X" );
 6:    strcpy( str2, "編集部です" );
 7:    printf( "1:[%s]¥n", str1 );
 8:    printf( "2:[%s]¥n", str2 );
 9:      /*s1にs2を連結*/
10:    strcat( str1, str2 );
11:    printf( "3:[%s]¥n", str1 );
12: }
```

リスト2-3：strlenのテスト(test2_3.c)

```
 1: void          main()
 2: {
 3:    char     str[32];
 4:    int      len;
 5:    strcpy( str, "ABCDEF" ); /*初期化*/
 6:       /*strの長さを得る*/
 7:    len=strlen( str );
 8:    printf( "1:[%d]¥n", len );
 9:       /*str[3]からの長さを得る*/
10:    len=strlen( &str[3] );
11:    printf( "2:[%d]¥n", len );
12: }
```

リスト2-4：strcmpのテスト(test2_4.c)

```
 1: void          main()
 2: {
 3:    char     str1[32], str2[32];
 4:    int      ret;
 5:       /*初期化*/
 6:    strcpy( str1, "ABC" );
 7:    strcpy( str2, "DEF" );
 8:     /*s1とs2を比較:ABC<DEF なので負*/
 9:    ret = strcmp( str1, str2 );
10:    printf( "1:[%d]¥n", ret );
11:    strcpy( str1, "ABC" );
12:    strcpy( str2, "@BC" );
13:      /*s1とs2を比較 : A>@ なので正*/
14:    ret = strcmp( str1, str2 );
15:    printf( "2:[%d]¥n", ret );
16:    strcpy( str1, "ABC" );
17:    strcpy( str2, "ABC" );
18:      /*s1とs2を比較 : 等しいので 0*/
19:    ret = strcmp( str1, str2 );
20:    printf( "3:[%d]¥n", ret );
21: }
```

リスト2-5：strchrのテスト(test2_5.c)

```
 1: char          *strchr();
 2: void          main()
 3: {
 4:    char     str[32], *ptr;
 5:       /*初期化*/
 6:    strcpy( str, "ABCDEFG" );
 7:       /*見つかったのでｱﾄﾞﾚｽを返す*/
 8:    ptr = strchr( str, 'D' );
 9:    printf( "1:[%s]¥n", ptr );
10:       /*見つからなかったのでNULL*/
11:    ptr = strchr( str, 'Z' );
12:    printf( "2:[%s]¥n", ptr );
13: }
```

リスト2-6：ブランクスキップ(test2_6.c)

```
 1: #include <stdio.h>
 2: void          skip_bk();
 3: void          main()
 4: {
 5:    int      i;
 6:    char     str[128];
 7:    printf( "文字列->" );
 8:    gets( str );          /*文字列入力*/
 9:    printf( "入力 -> [%s]¥n", str );
10:    skip_bk( str );
11:    printf( "変換 -> [%s]¥n", str );
12: }
13: void          skip_bk( str )
14: char          *str;
15: {
16:    int      i;
17:    /*前方のﾌﾞﾗﾝｸのｽｷｯﾌﾟ*/
18:    for( i=0; str[i]==' '; i++ ) ;
19:    /*ﾌﾞﾗﾝｸｽｷｯﾌﾟした後からｺﾋﾟｰ*/
20:    strcpy( str, &str[i] );
21:    /*文字列の長さを得る*/
22:    i = strlen( str );
23:    i--;
24:    /*後方のﾌﾞﾗﾝｸのｽｷｯﾌﾟ*/
25:    for(    ; str[i]==' '; i-- ) ;
26:    str[i+1]='¥0';     /*NULL付加*/
27: }
```

で出力ファイルはoutfileに変わり，

test3—3 < infile

で入力ファイルはinfileに変わります。後者ではHuman68kのtypeコマンドと同じです。では，簡単なコピーコマンドを作ってみましょう。書式は以下のようにします。

A>cpc 入力ファイル　出力ファイル

その前にコマンド行引数の取り込み方を説明します。これは，一般的にargc，argvという変数を使い，argcは引数の数，argvは引数自身です。では，argc，argvのテストをしてみましょう（**リスト3.4**）。

A>test3—4 hello world
argc = [3]
argv [0] = [A:¥TEST3—4.X]
argv [1] = [hello]
argv [2] = [world]

引数の数にはプログラム自身も含まれ，プログラムを起動したコマンドがargv [0]になります。さらに，K&Rには明示してありませんが，mainには3番目の引数envpもあります。これはプログラムが受け取る環境変数です。ここでは解説はせずに，表示方法のみ載せます。また，指定した環境変数の中身を見るgetenvという関数もあるので紹介します（**リスト3.5**）。

では，コピーコマンドを作ってみましょう。ここで注意してほしいのはエラー処理です。職業プログラマのプログラムではエラー処理が全体の半分以上を占めることも珍しくありません。特に，ファイルのオープンエラー処理などは絶対にチェックしたいものです。関数のリファレンスマニュアルにエラーリターン値が記述してある場合は，エラー処理は行ったほうがよいと思います。エラー表示には，**fprintf**を使います。fprintfは，画面以外のファイルにも出力でき，書式はprintfとほぼ同じです。

また，エラー処理を行ったりしたらプログラムが少し長くなりましたので，読み書きする場所を関数filecopyとして分割しました。filecopyには，引数としてファイルポインタを渡します。fputsのリターン値がEOFの場合は処理を中止していますが，これはディスクへの書き込みエラーが起こったことを示します。EOFは，マクロ（プリプロセッサ）定義されているものです。では，**リスト3.6**を，これは**cpc.c**として入力実行してください。

ところで，fgetsやfputsは，改行文字までを一度に読み込むものなので，ファイルから1文字単位で読み書きする場合には不適当です。そこで，1文字ずつ読み書きする**fgetc**，**fputc**の2関数を紹介します。それを利用した関数が**リスト3.7**です。これをリスト3.6のfilecopyの部分と差し替えてください。また，せっかく1文字ずつ読めるのですから，「もしwという文字が出たらメッセージを出して処理を中止する」ようにしてみましょう。wという文字が入っているファイルを入力ファイルとして実行してみてください。

バイナリファイルの例

ここで，さっき出てきたバイナリモード，テキストモードと＾Zについて，少し詳しく説明しましょう。Human68kやMS-DOSなどパソコンのOSの多くはテキストファイルの終わりに$1a_H$（＾Z）が入っています（ASCIIコードの1から26（$1a_H$）までは順に＾A～＾Zの制御文字に対応している）。よって，逆に$1a_H$が入っていることによりEOFが検出できるともいえます。つまり，なにかの拍子にテキストファイルの途中に＾Zが入ったら，そのあとはテキストモードでは内容を見ることはできなくなるわけです。これを，Human68kのエディタedで試してみましょう。以下のように，dummyという名前のファイルを作成してください。

```
このファイルはテキストです↓
ˢB↓
バイナリモードでしか読めません↓
```

ˢBの入力には，まず＾Vを入力し(これは次に入力する文字が制御文字であるということを意味する)次に＾Zを入力します。これを，type dummy としても1行目しか見ることはできません。リスト3.6で作ったコピーコマンドでも同じことです。つまり，バイナリモードでの読み書きが必要なのです。では，**リスト3.8**を実行してみてください。どうです，3行目まで見られたでしょう。これがバイナリモードです。

ただ，一般的にバイナリファイルとは何が入っているかわからないファイルです。よって文字列読み込み用のfgetsを使ったら変になる恐れもあります。そこで登場するのが，レコード長単位で読み書きする**fread/fwrite**です。では，リスト3.6のfilecopy部分を，**リスト3.9**に変更してください。これはファイルから256バイトずつ読み込むプログラムです。このとき，呼び出し側のfopenの第2引数“r”と“w”をそれぞれ“rb”と“wb”に変更することを忘れないでください。

ところで，freadの戻り値は実際に読み込んだバイト数です。つまり300バイトのファイルであれば，1回目は256バイト読めますが2回目は44バイトしか読めないわけです。256バイト読めと指定したにもかかわらず44バイトしか読めなかったらファイルの終わりかエラーなのです。また，fwriteでは実際に読んだ分しか書いてはいけません。44バイトしか読んでないのに256バイト書こうとしたら212バイトのゴミが書き込まれるからです。このプログラムでは実際に読み書きされたバイト数はreadnumという変数にセットしています。fread/fwriteは，一般的には固定長テキストファイルやバイナリファイルの操作に使用します。

ダイレクトアクセス

一般的に，ファイルの読み書きは時間がかかります。よって一気に1000バイト目などの読み書きができれば非常に重宝します。そこで，ファイルのダイレクトアクセスのために**fseek**，**ftell**などの関数が用意されて

リスト3

リスト3-1：ファイルオープン/クローズ(test3_1.c)

```
 1: #include <stdio.h>
 2: void          main()
 3: {
 4:     FILE      *fp;
 5:     fp=fopen( "data", "r" );
 6:     fclose( fp );
 7: }
```

リスト3-2：テキストファイルのコピー(test3_2.c)

```
 1: #include <stdio.h>
 2: void          main()
 3: {
 4:   FILE   *ifp, *ofp;
 5:   char    buff[256];
 6:   ifp=fopen( "test3_3.c", "r" );
 7:   ofp=fopen( "test3_3.out", "w" );
 8:   while(fgets( buff, 256, ifp )!=NULL)
 9:      fputs( buff, ofp );
10:   fclose( ifp );
11:   fclose( ofp );
12: }
```

リスト3-3：標準入出力間コピー(test3_3.c)

```
 1: #include <stdio.h>
 2: void          main()
 3: {
 4:   char          buff[256];
 5:   while(fgets( buff,256,stdin)!=NULL)
 6:      fputs( buff, stdout );
 7: }
```

リスト3-4：コマンド行引数の表示(test3_4.c)

```
 1: void          main( argc, argv )
 2: int argc;
 3: char          *argv[];
 4: {
 5:   int         i;
 6:   printf( "argc   =[%d]¥n", argc );
 7:   for( i=0; i<argc; i++ )
 8:   printf("argv[%d]=[%s]¥n",i,argv[i]);
 9: }
```

リスト3-5：環境変数の表示(test3_5.c)

```
 1: #include <stdlib.h>
 2: void          main( argc, argv, envp )
 3: int argc;
 4: char          *argv[];
 5: char          *envp[];
 6: {
 7:     /*全ての環境変数を見る*/
 8:  while( *envp++ )
 9:     printf( "%s¥n", *envp );
10:    /*環境変数 PATH を見る*/
11:  printf("PATH=[%s]¥n",getenv("PATH"));
12: }
```

います。では，edで以下の内容のファイルをdummy2として入力したあと，**リスト3.10**を実行してください。

ABCDEFGHIJKLMNOPQRSTUVWXYZ

ここで注意してほしいのは，2番目の引数オフセットの指定は，longで行うことです。XCではintも32ビットなのでintを使っても問題なく動きますが，offset＝10Lという形式で，long定数はLをつける習慣をつけたほうがいいと思います。

プログラムは，まず，ファイルの先頭から5Lバイト目へ移動し，1バイト読み表示します。すると6番目の文字であるFが表示されましたね。なぜ5バイト目なのに6番目の文字かというと，ファイルの先頭の文字Aは0バイト目だからです。次にftellで，6と表示されたのは，freadで1バイト読んでオフセットが1バイト進んだからです。これはエディタなどで1文字入力したらカーソルの位置が右に移動することと同じです。

ではファイル操作のまとめとして，簡単なファイルのダンプコマンドを作ってみました(**リスト3.11**)。ファイル関連の関数はここで紹介したものしか使っていません。プログラムは，ダンプ表示ファイル名を引数指定して起動します。そうして，表示開始アドレスと表示バイト数を入力します。

ここで初めて出てくるもので，#define SIZE 64などの表現があります。これはSIZEという文字列を64に置き換える**マクロ**定義です。少し，プリプロセッサがわかってきましたか？　またここで初めて出た関数で，sprintfとisprintがあります。sprintfは，fprintfのファイルポインタ部分が文字列に置き換わったもので，文字列に対して書式つきの出力を行う場合に使います。isprintは，引数が表示可能な文字かどうかを判断する関数です。Cのライブラリには，is～で始まる関数がたくさん用意されています。これらは非常に便利ですので，ぜひ試してください。このリストは，od.cとして入力してください。

構造体とポインタ

ここでは**構造体**の使い方について紹介します。構造体は，操作しやすいようにひとつの名前でまとめられた変数の集まりです。以下の例では，struct profileという型のデータ型を宣言していますが，PROFとtypedefしておきます。typedefとは，前もって(P)と宣言すれば，(D1)と宣言するのと(D2)と宣言するのは同じことになる機能です。

```
(P) typedef宣言
    typedef struct tag {
        char a [10];
        int  b;
    } TAG;
(D1) 構造体変数の宣言1
    struct tag val;
(D2) 構造体変数の宣言2
    TAG val;
```

C言語におけるtypedef機能は非常に便利なもので，入出力で出てきたFILEも実は構造体をtypedefしてあるものなのです。では，ヘッダファイルprofile.h（**リスト4.1**）を入力してください。

次に，本来のソース（**リスト4.2**）です。ここで，いま入力したヘッダファイルprofile.hをインクルードすることを忘れないでください。注意したいのは，通常の<>ではなく“”で囲むことです。とりあえず，システムから提供されたものは< >で，自分で作ったものは“”で囲むと考えていいでしょう。

どうです，簡単でしょう。構造体なんて。構造体の中の各要素のアクセスには.(ピリオド)を使います。つまりjiroのtel(二郎の電話番号)を見るにはjiro.telとすればいい

リスト3-6：コピーコマンド第1版(cpc.c)

```
 1: #include <stdio.h>
 2: void         filecopy();
 3: void         main(argc, argv)
 4: int argc;
 5: char   *argv[];
 6: {
 7:    FILE     *ifp, *ofp;
 8: 
 9:    if( argc<3 ){
10:       fprintf( stderr, "使用方法:CPC 入力ﾌｧｲﾙ 出力ﾌｧｲﾙ¥n");
11:       return;
12:    }
13:    if(( ifp=fopen( argv[1], "r" )) == (FILE*)NULL ){
14:       fprintf( stderr, "%sがｵｰﾌﾟﾝできません¥n", argv[1] );
15:       return;
16:    }
17:    if(( ofp=fopen( argv[2], "w" )) == (FILE*)NULL ){
18:       fprintf( stderr, "%sがｵｰﾌﾟﾝできません¥n", argv[2] );
19:       return;
20:    }
21:    filecopy( ifp, ofp );
22:    fclose( ifp );
23:    fclose( ofp );
24: }
25: void         filecopy( ifp, ofp )
26: FILE         *ifp, *ofp;
27: {
28:    char       buff[256];
29: 
30:    while( fgets( buff, 256, ifp )!=NULL ){
31:       if( fputs( buff, ofp )== EOF ){
32:          fprintf( stderr, "ﾌｧｲﾙ書き込みｴﾗｰ¥n" );
33:          break;
34:       }
35:    }
36: }
```

リスト3-7：コピーコマンド第2版(差し替え部分)

```
 1: void         filecopy( ifp, ofp )
 2: FILE         *ifp, *ofp;
 3: {
 4:   int        c;
 5:   while(( c=fgetc( ifp ))!=EOF ){
 6:     if( fputc( c, ofp )== EOF ){
 7:       fprintf(stderr,"書き込みｴﾗｰ¥n");
 8:       break;
 9:     }
10:     else if( c=='w' ){
11:       fprintf(stdout,"[w]があった¥n");
12:       break;
13:     }
14:   }
15: }
```

リスト3-8：バイナリファイルの表示(test3-8.c)

```
 1: #include <stdio.h>
 2: void         main()
 3: {
 4:   FILE       *fp;
 5:   char        buff[256];
 6:      /* bｵﾌﾟｼｮﾝでｵｰﾌﾟﾝ */
 7:   fp=fopen( "dummy", "rb" );
 8:   while( fgets( buff, 256, fp )!=NULL)
 9:      fputs( buff, stdout );
10: }
```

リスト3-9：コピーコマンド第3版(差し替え部分)

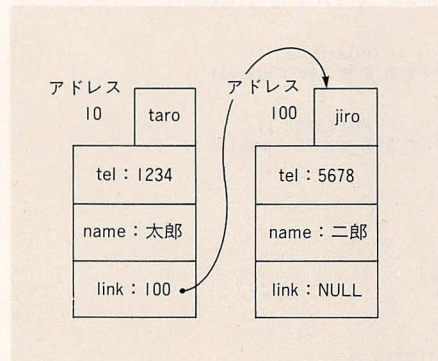
```
 1: void         filecopy( ifp, ofp )
 2: FILE         *ifp, *ofp;
 3: {
 4:   char  buff[256];
 5:   int   readnum=256;
 6:   while( readnum==256 ){
 7:     readnum=fread( buff,1,256,ifp);
 8:     if( fwrite( buff,1,readnum,ofp )
 9:                           !=readnum ){
10:       fprintf(stderr,"書き込みｴﾗｰ¥n");
11:       break;
12:     }
13:   }
14: }
```

リスト3-10：ファイルのダイレクトアクセス(test3_10.c)

```
 1: #include <stdio.h>
 2: void         main()
 3: {
 4:    FILE   *fp;
 5:    int           buff;
 6:    long          offset=5L;
 7:    fp=fopen( "dummy2", "rb" );
 8:    fseek( fp, offset, 0 );
 9:    printf( "offset=[%ld]¥n", offset );
10:    fread((int*)&buff, 1, 1, fp );
11:    printf( "read=[%c]¥n",  buff[0] );
12:    offset = ftell( fp );
13:    printf( "offset=[%ld]¥n", offset );
14:    fclose( fp );
15: }
```

のです。もし，変数がポインタであれば.の代わりに−>を使えばいいだけです。よって，**リスト4.3**で構造体とポインタの使い方を紹介します。ところで，リスト4.2では変数として宣言しましたが，今度はポインタとしての宣言のみしかしません。よって，ポインタの数だけ実際の構造体のエリアを確保しなければいけません。そこで出てくるのが，動的にメモリを確保する関数mallocです。確保したいサイズを引数として渡し，プログラム中にメモリを割り当てることができます。sizeof (PROF) と書いてあるのはデータ型PROFの大きさだけメモリを確保するというものです。

実行したら，taro−>linkの値とjiroの値が同じことがわかるでしょう。これを，図解すると**図3**のようになります。つまりポインタがアドレスを持っており，その指す先に構造体のデータがあるわけです。

図3　構造体とポインタ

構造体を使ってみよう

次に，すでに定義されている構造体の使い方を紹介しましょう。いままでは自分で定義した構造体を使っていましたが，ここでは前もって準備された構造体を使ってみましょう。例題として，ファイルの情報を得るプログラムを作ってみましょう。これには**stat**という関数を使います。

```
#include <stat.h>
int    stat ( path, buf )
char  *path;
struct stat  *buf;
```

ファイル名を渡せば，struct stat型の変数bufに以下の情報がセットされます。

st_mode	：ファイル属性 (ディレクトリか通常ファイルか)
st_dev	：ドライブ番号
st_size	：ファイルの大きさ
st_atime	：ファイルの最終更新日時

ところで，XCのマニュアルには，この各メンバの属性についての解説がありません。今回，初めてこの関数を使いましたが困りました。とりあえずstat.hの中を見て書きましたが，本来インクルードファイルの中はユーザーには公開されていないのです。また，これらはすべて元の数値情報なので見やすい形式に直す必要もあります。そこで，新たに構造体f—infoを定義してそこにデータのセットを行います。

st—modeは，S—IFDIRかS—FREGビットを指定するとあります。これは，S—IFDIRでビットごとの論理積(&)を取り，真だったらディレクトリだということです。st—devのドライブ番号は，0だったらAドライブ1だったらBドライブ……となっていくので，値にAを足せばいいですね。st—atimeは，実は1970年1月1日0時0分0秒からの秒数なのです。よってこれは関数ctimeによって普通の表現に変えましょう。このためにはtime.hをインクルードしてください。とりあえずディレクトリだったらメッセージを出して終了するようにします。リスト1行目の#ifdef MSDOSは，MS-DOSの場合は2，3行目が有効になるプリプロセッサ命令です(MS-Cver5ではすでに定義してあるので定義不要)。では**リスト4.4**をstat.cという名前で入力してください。

プリプロセッサ

さて，この原稿のいたるところに，まるで呪文のようにプリプロセッサと出てきました。今回のC言語特集は入門編ですのでプリプロセッサ命令についてはあまり解説しません。しかし，プリプロセッサ命令は非常に便利な機能ですので，その機能を多用したラインエディタedl.cを作ってみまし

リスト3-11：ファイルのダンプコマンド(od.c)

```
 1: #include   <ctype.h>
 2: #include   <stdio.h>
 3: #define     SIZE    64        /*マクロ定義*/
 4: #define     TRUE    1         /*マクロ定義*/
 5: #define FALSE       0         /*マクロ定義*/
 6: void         dumpadr();
 7: void         main( argc, argv )
 8: int          argc;
 9: char         *argv[];
10: {
11:    FILE   *fp;
12:    long    start;/*開始アドレス*/, length;/*表示範囲*/
13:
14:    if( argc != 2 ){
15:       fprintf( stderr, "使用方法 od ファイル名¥n" );
16:       return;
17:    }
18:    if(( fp=fopen( argv[1], "rb" )) == (FILE*)NULL ){
19:       fprintf( stderr, "%sがオープンできません¥n", argv[1] );
20:       return;
21:    }
22:    while(1){
23:       if( !getadr( &start, &length ))break;
24:       dumpadr( fp, start, length );
25:    }
26:    fclose( fp );
27: }
28: int getadr( start, length ) /*開始アドレスと範囲の入力*/
29: long   *start, *length;
30: {
31:    char     buf[SIZE], *dummy;
32:
33:      /*先頭に0が付けば8進、0xが付けば16進、それ以外は10進*/
34:    printf( "¥n入力( 8進:0??, 10進:??, 16進:0x?? )¥n");
35:    printf( "¥t¥t¥t[終了 : q ]¥n" );
36:    printf( "スタートアドレス   ->" );                /*開始アドレス入力*/
37:    fgets( buf, SIZE, stdin );                            /*入力*/
38:    if( buf[0]=='q' ) return( FALSE );    /*'q'入力 : 0リターン*/
39:    *start = strtol( buf, dummy, 0 ); /*文字列をlongに変換*/
40:    printf( "ダンプバイト数 ->" );              /*表示バイト数入力*/
41:    fgets( buf, SIZE, stdin );                            /*入力*/
42:    if( buf[0]=='q' )return( FALSE );      /*'q'入力 : 0リターン*/
43:    *length = strtol( buf, dummy, 0 );/*文字列をlongに変換*/
44:    return( TRUE );               /*正しい指定がされた : 1リターン*/
45: }
46: void   dumpadr( fp, start, length )/*start→lengthの表示*/
47: FILE  *fp;
48: long         start;
49: long         length;
50: {
51:    long     pos;                /*現在のファイルポインタ*/
52:    int      ct;                 /*カウンタ*/
53:    int      readbuf;            /*読み込み用バッファ*/
54:    char     dispbuf[81];    /*表示用バッファ*/
55:
56:    if(( fseek( fp, start, 0 )) != 0 ){      /*startへseek*/
57:        fprintf( stderr, "シークエラー %ld(%lx)¥n", start, start);
58:        return;
59:    }
60:    for( pos=start; pos<start+length;  ){
61:       sprintf( dispbuf, "%80s", "" );/*buf初期化:空白詰め*/
62:       sprintf( dispbuf, "%08x", pos );           /*アドレス表示*/
63:       dispbuf[8]=' ';                           /*NULLを消す*/
64:       for( ct=0; ct<16; ct++ ){
65:            /*1バイト読み込み : readbufはintに型変換する*/
66:          if( fread((char*)&readbuf, 1, 1, fp )!=1 ){
67:             printf( "%s¥n", dispbuf );  /*EOF : 残りを出力*/
68:             return;                                 /*終了*/
69:          }
70:            /*16進表示部分 : 0?の形式で表示*/
71:          sprintf( &dispbuf[10+ct*3], "%02x", readbuf );
72:          dispbuf[12+ct*3] = ' ';                 /*NULLを殺す*/
73:            /*文字表示部分: 表示できない文字は'.'に変換*/
74:          if( !isprint( readbuf ))readbuf='.';
75:          dispbuf[60+ct]=readbuf;
76:          pos ++;
77:          if( pos>=start+length )break;
78:       }
79:       printf( "%s¥n", dispbuf );           /*dispbufの表示*/
80:    }
81:    return;
82: }
```

た。このプログラムはファイルの指定行を編集するラインエディタです。ファイル名を指定して起動します。

ここでは，エスケープシーケンスを表示する機能を関数もどきとしてマクロ定義しています（Human68kのマニュアルの最後に出てくる）。マクロは，単純だが入力ミスが起きやすい部分（'¥n'と'¥0'など）に有効です。'¥0'がNULLと定義してあるように'¥n'をNLなどと定義するのもいいでしょう。また，条件付きコンパイル（リスト4.4の#ifdef MSDOSの例）を使えば，異なったマシンでも動くプログラムの開発が容易になります。なおこのエスケープシーケンスは，MS-DOSと共通のものを使いましたのでMS-DOSでも正常に動きます。

リストの解説は行わないので，独自に解析してください。line—edit関数に編集行の文字列と長さを渡します。エディタで有効なキーは以下のようになっています。

キー	：機能
^R	：カーソルを右に移動
^L	：カーソルを左に移動
^I	：1文字空白を挿入
^D	：1文字削除
ESC	：終了

なおこのエディタはタブや漢字が入っていたら正常に動きません。また，編集できる最大のカラム数は80です。**リスト5.1**を，**edl.c**として入力してください。

プロセス管理

では，最後に演習としていままで習ったことを1本のプログラムにしてみましょう。ファイル名を指定して，メニューで以下のことを行うというものです。

1：ファイル情報表示	(stat	：リスト4.4)
2：ファイル内容表示	(cpc	：リスト3.6)
3：ファイルのダンプ	(od	：リスト3.11)
4：ファイルの編集	(edl	：リスト5.1)
5：子プロセス起動	(command	：command.x)

そうです，わざわざtest?—?という名前以外で入力したのは，ここで再利用するためだったのです。1～4は，いままでに作ったプログラムでそれぞれロードモジュールがディスクにあると思います。これを関数ライクに呼び出すのです。Cでは，子プロセスの生成，つまりプログラムの中から別のプログラムの起動が容易にできます。元来プログラムは，OS（基本ソフトウェア）が起動しているもので，wpやedなどのHumanのプログラム，そうしてコマンドインタプリタ（Human68kではCOMMAND.X，MS-DOSではCOMMAND.COM，UNIXではcshやsh）もOSが起動しているのです。そして，起動されたプログラムは別のプログラムを起動することもできるのです。

ここで初めて出てくる関数はsystemです。これは，引数として子プロセスとして起動するコマンド文字列を与えます。execという関数もありますが，書式が複雑になるので今回はsystemを使います。Human68kのコマンドをプログラムの中から引数

リスト4-1：構造体の定義(profile.h)

```
1:    typedef  struct   profile        {
2:       char     name[20];       /*名前*/
3:       int      tel;        /*電話番号*/
4:       char     adr[80];        /*住所*/
5:       struct   profile  *link; /*ﾘﾝｸ*/
6:    }PROF;   /*ﾌﾟﾛﾌｧｲﾙ*/
```

リスト4-2：構造体と変数(test4-2.c)

```
 1: #include <stdio.h>
 2: #include "profile.h"
 3: void          main()
 4: {
 5:    PROF     taro, jiro;
 6:    /*変数taroにﾃﾞｰﾀｾｯﾄ*/
 7:    strcpy( taro.name, "太郎" );
 8:    taro.tel=1234;
 9:    strcpy( taro.adr,"千代田区九段下");
10:    /*変数jiroにﾃﾞｰﾀｾｯﾄ*/
11:    strcpy( jiro.name, "二郎" );
12:    jiro.tel=9876;
13:    strcpy( jiro.adr, "中野区中野" );
14:    /*taroとjiroのデータ表示*/
15:    printf( "[%s][%d][%s]¥n",
16:      taro.name, taro.tel, taro.adr );
17:    printf( "[%s][%d][%s]¥n",
18:      jiro.name , jiro.tel, jiro.adr );
19: }
```

リスト4-3：構造体とポインタ(test4-3.c)

```
 1: #include <stdio.h>
 2: #include "profile.h"
 3: char         *malloc();
 4: void         main()
 5: {
 6:    /*ﾎﾟｲﾝﾀ変数の宣言*/
 7:   PROF      *taro, *jiro;
 8:    /*taroの分の領域確保*/
 9:   if(( taro = (PROF*)malloc( sizeof( PROF))) == (PROF*)NULL ){
10:     fprintf( stderr,"ﾒﾓﾘ確保失敗¥n" );
11:     return;
12:   }
13:      /*変数taroにﾃﾞｰﾀｾｯﾄ*/
14:   strcpy( taro->name, "太郎" );
15:   taro->tel=1234;
16:   strcpy( taro->adr,"千代田区九段下");
17:      /*jiroの分の領域確保*/
18:   if(( jiro = (PROF*)malloc( sizeof( PROF))) == (PROF*)NULL ){
19:     fprintf( stderr,"ﾒﾓﾘ確保失敗¥n" );
20:     return;
21:   }
22:     /*taroのlinkにjiroのｱﾄﾞﾚｽｾｯﾄ*/
23:   taro->link=jiro;
24:     /*変数jiroにﾃﾞｰﾀｾｯﾄ*/
25:   strcpy( jiro->name, "二郎" );
26:   jiro->tel=9876;
27:   strcpy( jiro->adr, "中野区中野" );
28:     /*jiroのlinkにNULLｾｯﾄ*/
29:   jiro->link=(PROF*)NULL;
30:     /*表示*/
31:   printf( "[%d][%s][%d][%s][%d]¥n",
32:     taro, taro->name , taro->tel, taro->adr, taro->link );
33:   printf( "[%d][%s][%d][%s][%d]¥n",
34:     jiro, jiro->name , jiro->tel, jiro->adr, jiro->link );
35: }
```

リスト4-4：ファイル属性の取り出し(stat.c)

```
 1: #ifdef MSDOS          /* MS-DOS(MSCver5)*/
 2: #include <sys/types.h>
 3: #include <sys/stat.h>
 4: #else                 /* Human68K */
 5: #include <stat.h>
 6: #endif
 7: #include <time.h>
 8: #include <stdio.h>
 9: #define SIZE 256
10: void     main(argc, argv)
11: int      argc;
12: char     *argv[];
13: {
14:    typedef  struct    f_info{
15:      char   drive;          /*ﾄﾞﾗｲﾌﾞ番号*/
16:      long   filesize;         /*ﾌｧｲﾙｻｲｽﾞ*/
17:      char   filedate[SIZE]; /*更新日時*/
18:    }F_INFO;
19:    F_INFO   file;                  /*変換バッファ*/
20:    struct   stat     buff;   /*stat構造体用バッファ*/
21:
22:    if( argc != 2 ){
23:       fprintf( stderr, "使用方法 stat ﾌｧｲﾙ名¥n" );
24:       return;
25:    }
26:    if( stat( argv[1], &buff )!= 0 ){
27:       fprintf( stderr, "%sが見つかりません¥n", argv[1] );
28:       return;
29:    }
30:       /*ﾌｧｲﾙの種類:ﾃﾞｨﾚｸﾄﾘは除外*/
31:    if( buff.st_mode & S_IFDIR ){
32:       printf( "このﾌｧｲﾙはﾃﾞｨﾚｸﾄﾘです¥n" );
33:       return;
34:    }
35:       /*ﾄﾞﾗｲﾌﾞ番号 0:A 1:B....*/
36:    file.drive = buff.st_dev+'A';
37:    printf( "ﾄﾞﾗｲﾌﾞ:%c¥n", file.drive );
38:      /*ﾌｧｲﾙｻｲｽﾞ  :ﾊﾞｲﾄ*/
39:    file.filesize = buff.st_size;
40:    printf( "ｻｲｽﾞ   :%ld ﾊﾞｲﾄ¥n", file.filesize );
41:      /*ﾌｧｲﾙの最終更新日時*/
42:    strcpy( file.filedate, ctime( &buff.st_atime ));
43:    printf( "日時   :%s¥n", file.filedate );
44: }
```

つきで実行することも可能なのです。

「だったら，そんなものばかり組み合わせてプログラム作ればいいじゃないか！」といったあなたは正しい。そう，それも広い意味での構造化なのです。実際，UNIXのソースには，いたるところにforkとexecというプロセス制御用の関数が出てきます。**リスト6.1**をftool.cという名前で入力してください。

プログラム上達には

文法で，＝と＝＝，'¥0'と'¥n'などは似ているのにまったく異なったものです。これらのミスはコンパイラでは発見できないので注意しましょう。UNIXのlintみたいなものがあればそれでプログラムのチェックもやってなるべく「清く正しい」プログラムを作るように心がけたほうがいいと思います。それが，結局，時間の短縮につながります。C言語に限らずプログラムの上達のコツは①たくさん作ってみる，②わからないことは人に聞くの2点です。これはプログラムに限らないのかもしれませんが，とにかく作ってみることです。そうしてわからないところは聞くことです。みんなで「清く正しい」Cプログラマを目指しましょう。

リスト5-1：ラインエディタ(edl.c)

```
 1: #include <stdio.h>
 2: #include <ctype.h>
 3: #define RIGHT   '¥x12'         /*^R:ｶｰｿﾙ右移動はｺﾝﾄﾛｰﾙR */
 4: #define LEFT    '¥x0c'         /*^L:ｶｰｿﾙ左移動はｺﾝﾄﾛｰﾙL */
 5: #define DEL     '¥x04'         /*^D:文字削除はｺﾝﾄﾛｰﾙD */
 6: #define INS     '¥x09'         /*^I:文字挿入はｺﾝﾄﾛｰﾙI */
 7: #define ESC     '¥x1b'         /* ESC:終了はESC */
 8: #define reset()        printf("¥033[0m")       /*画面ﾘｾｯﾄ*/
 9: #define reverse()      printf("¥033[7m")       /*画面反転*/
12: #define clr_screen()   printf("¥033[2J")        /*画面ｸﾘｱ*/
10: #define csr_right(co)  printf("¥033[%dC",co) /*ｶｰｿﾙ右移動*/
11: #define csr_left(co)   printf("¥033[%dD",co) /*ｶｰｿﾙ左移動*/
13: #define csr_mov(li,co) printf("¥033[%d;%dH",li,co) /*移動*/
14: #define color()        printf("¥033[32m")        /*色表示*/
15: #define SIZE 81     /*1行の最大ｻｲｽﾞは80ﾊﾞｲﾄ*/
16: void        line_edit();
17: void        main(argc, argv)
18: int argc;
19: char   *argv[];
20: {
21:     char    buf[SIZE];
22:     FILE    *fp;
23:     int     line, i=1;
24:     long    size;
25:
26:             if( argc != 2 ){
27:        fprintf( stderr, "使用方法 edl ﾌｧｲﾙ名¥n" );
28:        return;
29:             }
30:     if(( fp=fopen( argv[1], "r+" )) == (FILE*)NULL ){
31:        fprintf( stderr, "%sがｵｰﾌﾟﾝできません¥n", argv[1] );
32:        return;
33:     }
34:     printf( "編集行?->" );  /*行の入力*/
35:     scanf( "%d", &line );   /*行の入力*/
36:     while( fgets( buf, SIZE, fp )!=NULL )
37:        if( i++ == line )break;/*指定行までｽｷｯﾌﾟ*/
38:     size=strlen(buf); buf[size-1]='¥0';/*行の長さを得る*/
39:     clr_screen();
40:     color();                              /*色を変える*/
41:     csr_mov(8,3);                         /*8行目の3ｶﾗﾑ目に移動*/
42:     printf( "%d行目の編集です¥n", line );
43:     csr_mov(15,1);                       /*15行目の1ｶﾗﾑ目に移動*/
44:     reset();
45:     printf("¥n¥t ^R  : ｶｰｿﾙを右に移動¥n" );
46:     printf("¥t ^L  : ｶｰｿﾙを右に移動¥n" );
47:     printf("¥t ^I  : 1文字空白を挿入¥n" );
48:     printf("¥t ^D  : 1文字削除¥n" );
49:     printf("¥t ESC : 終了¥n" );
50:     csr_mov(10,1);                       /*10行目の1ｶﾗﾑ目に移動*/
51:     line_edit( buf, size-1 );         /*行編集*/
52:     if(( fseek( fp, -(size+1), 1 )) != 0 ){/*ﾌｧｲﾙ書き込み*/
53:        fprintf( stderr, "ｼｰｸｴﾗｰ %s¥n", argv[1] );
54:        return;
55:     }
56:     fprintf( fp, "%s¥n", buf );
57:     fclose(fp);
58: }
59: void         line_edit( buf, length )
60: char         *buf;
61: int length;
62: {
63:     int   ch=' '/*読込文字*/, csrpos=0/*ｶｰｿﾙ位置*/, i;
64:
65:     reverse();
66:     printf( "%s", buf );
67:     csr_left(length+1);
68:     while((ch = getch())!= ESC ){
69:        if( ch == RIGHT && csrpos < length-1 ){
70:           csrpos++;
71:           csr_right(1);
72:        }
73:        else if( ch == LEFT && csrpos > 0 ){
74:           csrpos--;
75:           csr_left(1);
76:        }
77:        else if( ch == DEL && csrpos < length ){
78:           for( i=csrpos; i < length-1; i++ )
79:              buf[i] = buf[i+1];
80:           buf[length-1] = ' ';
81:              printf( "%s", &buf[csrpos] );
82:           csr_left(length-csrpos);
83:        }
84:        else if( ch == INS && csrpos < length ){/**/
85:           for( i=length-1; i>csrpos; i-- )
86:                      buf[i] = buf[i-1];
87:           buf[i] = ' ';
88:              printf( "%s", &buf[csrpos] );
89:           csr_left(length-csrpos);
90:        }
91:        else if( isprint(ch) && csrpos < length ){
92:           buf[csrpos] = ch;
93:           printf( "%c", ch );
94:           if( csrpos < length-1 ) csrpos++;
95:           else csr_left(1);
96:        }
97:           }
98:     reset();
99: }
```

リスト6-1：子プロセスの実行(ftool.c)

```
 1: #include <stdlib.h>
 2: #include <stdio.h>
 3: void          main( argc, argv )
 4: int argc;
 5: char          *argv[];
 6: {
 7:    FILE       *fp;
 8:    int         c=' ';
 9:    char        buf[36];
10:
11:    if( argc<2 ){
12:       fprintf( stderr, "使用方法: ftool  ﾌｧｲﾙ名¥n" );
13:       return;
14:    }
15:    if(( fp=fopen( argv[1], "r" )) == (FILE*)NULL ){
16:       fprintf( stderr, "%sがｵｰﾌﾟﾝできません¥n", argv[1] );
17:       return;
18:    }
19:    fclose( fp );
20:    while( 1 ){
21:       printf( "0:終了¥n" );
22:       printf( "1:ﾌｧｲﾙ情報表示¥n" );
23:       printf( "2:ﾌｧｲﾙ内容表示¥n" );
24:       printf( "3:ﾌｧｲﾙﾀﾞﾝﾌﾟ¥n" );
25:       printf( "4:ﾌｧｲﾙ編集¥n" );
26:       printf( "5:子ﾌﾟﾛｾｽ起動(EXITで終了)¥n" );
27:       c=getch();
28:       if( c=='0' )
29:          break;
30:       else  if( c=='1' ){
31:          strcpy( buf, "stat " );
32:          strcat( buf, argv[1] );
33:       }
34:       else  if( c=='2' ){
35:          strcpy( buf, "cpc " );
36:          strcat( buf, argv[1] );
37:          strcat( buf, " con" );
38:       }
39:       else  if( c=='3' ){
40:          strcpy( buf, "od " );
41:          strcat( buf, argv[1] );
42:       }
43:       else  if( c=='4' ){
44:          strcpy( buf, "edl " );
45:          strcat( buf, argv[1] );
46:       }
47:       else  if( c=='5' ){
48:          strcpy( buf, "command" );
49:       }
50:       else{
51:          printf("指定が間違ってます %c¥n", c );
52:          continue;
53:       }
54:       if( system( buf )!= 0 ){
55:          fprintf( stderr, "ｺﾏﾝﾄﾞ起動に失敗しました¥n" );
56:       }
57:    }
58: }
```

●特集　Cプログラミングへの招待

使うための基礎知識
C言語実戦マニュアル

Nakamori Akira
中森　章

C言語をC言語らしく使うことは，データ型を駆使することです。それには実際にどのようにメモリに格納されるのかということを知っていなければなりません。ここではマシン語の知識を基にCの文法を再検討してみましょう。

Cは単純な言語です。実際，C言語を学ぶ際に知っておかなければならないのは，データ型と変数の宣言，関数の作り方，式と演算子，制御構造くらいしかありません。これにプリプロセッサとポインタと構造体の知識があればほぼ完璧です。

その半面，C言語は高級アセンブラといわれ，アセンブリ言語の知識がないと理解するのが難しいとされています。しかし，C言語に要求されるアセンブリ言語の知識とはニーモニックや疑似命令といった直接的なものではなく，アセンブリ言語をどのように使ってプログラムを書けばいいのかという概念的なものです。言語仕様自体は単純なC言語ですが，ポインタや構造体を理解しづらいと感じる人が多いのはこの知識が欠けているためだと思われます。

以下では，必要となるアセンブリ言語の知識を適当に盛り込みながら，C言語を理解するうえでどうしても知っておかなければならない，変数，演算子，関数，ポインタ，構造体および共用体の考え方について説明していきたいと思います。

データ型と変数の宣言

C言語に限らずプログラミング言語の目的はデータを処理することです。どのようなデータを扱えるかで，その言語の特徴が決まるといっても過言ではありません。

XCを始めとする標準的なC言語で用意されている基本的なデータ型は表1に示す10種類です。unsigned（符号なし）とかshort（短い）とかlong（長い）とか形容詞がついていますが基本的にはchar型，int型，float型およびdouble型にまとめることができます。charはcharacter（文字），intはinteger（整数），floatはfloating point（浮動小数点），doubleはdouble precision（倍精度）に由来する名称です。

●整数

整数のうちint型はC言語のプログラムを実行するマシンのCPUの“自然な”長さ（ビット長）を持つ整数に相当します。つまり，16ビットCPUならばint型は16ビット長，32ビットCPUならばint型は32ビット長です（8ビットではint型は16ビットだったが)。そして，int型に対するshortとかlongという飾りはその自然な整数と比べて短いか長いかということを示しているのです。

いわゆる32ビットCPU（68000を含む）では8ビット，16ビット，32ビットという単位で整数を扱えるようになっていますから，

　8ビット整数　→　char
　16ビット整数　→　short int
　32ビット整数　→　int

は実に自然な対応です。これならlong int型は64ビット整数のはずですが，現在の処理系ではint型と同じ32ビットとなっています（64ビット整数なんてまず使わない)。

●浮動小数点数

常識的にはfloat型（単精度）は32ビット長，double型（倍精度）は64ビット長です。ただし，C言語では倍精度は単精度よりも精度が落ちなければいいと決められているだけです（同じ精度でもいい)。なおANSI規格ではlong double型（拡張精度）も存在するようですが，これは80（または96）ビット長になるのではないでしょうか。

変数の宣言

なにかのデータを扱うつもりなら，そのデータを格納するための変数を宣言しなければなりません。変数の宣言は，

　データ型名　変数名；

という形式で行います。また，同じデータ型の変数をまとめて宣言する場合は，

　データ型名　変数名，変数名，……；

というように変数名をカンマ(,)で区切って並べます。たとえば，

　double rx78，msz006；

という宣言はrx78，msz006という2つのdouble型の変数の宣言を意味します。

このとき，メモリにはそれぞれの変数を格納するのに必要な領域（この場合は8バイト領域が2つ）が確保され，その先頭アドレスにはrx78, msz006と1対1に対応するラベルがつけられます（図1)。

変数の実体がメモリにあるということは考えてみれば当然ですが，ほかの言語ではこれを特に意識する必要はありません。しかし，のちに説明するアドレス演算子（&）やポインタは変数がメモリにあることを前提としていますから，C言語ではこのことをはっきりと意識しておかねばなりません。

配列の宣言

変数が宣言されるとその実体はメモリに割り当てられます。しかし，

　int　rx78，rx178，rx78nt1；

図1　変数の格納状態

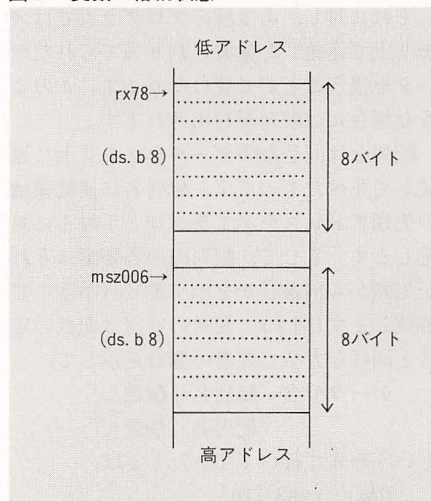



表1　基本的なデータ型(XCの場合)

	型	ビット長	範囲
整数	unsigned char	8	0～255
	char	8	−128～127
	unsigned int	32	0～4298967295
	int	32	−2147483648～2147483647
	unsigned short int	16	0～65535
	short int	16	−32768～32767
	unsigned long int	32	0～4298967295
	long int	32	−2147483648～2147483647
実数	float	32	約±10^{-37}～10^{38}
	double	64	約±10^{-307}～10^{308}

図2　配列の様子

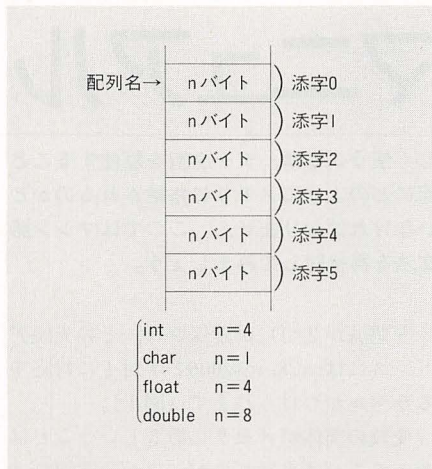
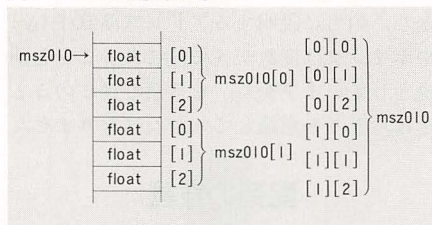



図3　2次元配列の考え方

という変数宣言を行った場合，同時に宣言したrx78，rx178，rx78nt1に対応する領域がメモリ上で連続した領域に割り当てられるということは保証されていません。

それに対し，ある種のプログラムではメモリ上で連続した領域に割り当てられたデータを扱うことが必要になります。このような場合には配列が利用されます。

配列とは同じ型のデータをメモリ上に連続して並べたものです。配列名は連続領域の先頭アドレスを示すラベルと1対1に対応します。そして，配列内の各要素はそれが先頭から何番目かを示す番号（添字）で参照します(図2)。配列の宣言は変数の宣言と同様な方法で要素の個数を示して，

　　データ型名　配列名［個数］，
　　　　　　　　配列名［個数］，……；

という形式で行います。たとえば，

　　float　rx93［20］；

という宣言は，

　　rx93［0］，rx93［1］，…，rx93［19］

という20個のfloat型要素を持つ配列の宣言を意味します。

2次元配列の場合は，

　　データ型名　配列名［個数］［個数］；

によって宣言することができます。

　　float　msz010［2］［3］；

という宣言は，

　　msz010［0］［0］，msz010［0］［1］，
　　msz010［0］［2］，
　　msz010［1］［0］，msz010［1］［1］，
　　msz010［1］［2］

という6個のfloat型要素を持つ配列の宣言を意味します。一般に，

　　データ型名　配列名［n］［m］；

という宣言はn行m列の形をしたn×m行列の宣言に等しくなります。また，これはm個の要素を持つ1次元配列を単位要素とするn個の要素を持つ1次元配列とみなすこともできます。たとえば，

　　float msz010［2］［3］；

という宣言では，msz010は2個の要素，

　　msz010［0］，msz010［1］

を持つ1次元配列とみなすことができます。この［0］，［1］と添字のついた2つの要素はfloat型ではありません。それぞれが3個のfloat型を要素とする1次元配列なのです。そして，

　　msz010［0］，msz010［1］

という名前は2組の1次元配列の先頭，すなわち，

　　msz010［0］［0］，msz010［1］［0］

という要素のアドレスに対応しています。この考えで，

　　msz010［1］［2］

という表現を眺めると，これは，

　　(msz010［1］)［2］

と同じであり，1次元配列（msz010［1］）の3番目（添字が2だから）の要素を示すということがわかります(図3)。通常2次元配列は縦横の表とか行列といったイメージでとらえられることが多いのですが，このように1次元配列を要素とする1次元配列とみなすことによって，また違った理解が得られるのではないでしょうか。

ところで，C言語では3次元，4次元，それ以上の次元の配列も考えることができます。3次元配列は1次元配列を要素とする2次元配列，4次元配列は1次元配列を要素とする3次元配列と考えていけば次元が増えても恐れることはありません。

変数の初期化

C言語では変数を宣言すると同時に，その変数の初期値を設定することができます。ちょっとアセンブリ言語を思い出してください。アセンブリ言語ではメモリ領域を確保するための疑似命令として，ただ領域を確保する命令（68000ではds.b, ds.w, ds.l）と値を入れて領域を確保する命令(68000ではdc.b, dc.w, dc.l)があります。変数の初期化は値を入れてメモリ領域を確保する命令に相当します。

変数の初期化は，変数の宣言のときに変数名の後ろに＝をつけて値を指定することで行います。たとえば，

　　float rx78＝1.2, rx178, rx78nt1＝3.14;

という宣言ではrx78には1.2という初期値，rx78nt1には3.14という初期値を与えることを意味しています。このときrx178の初期値は記憶クラスにより0または不定です。

初期値の設定が特に有用になるのは配列を初期化する場合です。配列はプログラム中で参照する数値テーブルなどとして使われることが多く，しかもその要素数が多いので，初期化をプログラムで行っていたのでは貴重な実行時間が無駄になってしまいます。また，プログラムを読みやすくするためには，数値テーブルなどは変数宣言時に初期化しておくほうがいいと思われます。

配列の初期化は変数の初期化と同様に＝によって行いますが，この場合は要素が複数になりますからまとまりをはっきりさせるために{と}で初期値をカンマ(,)で区切って囲みます。たとえば，

　　int msz006［4］＝ {1, 2, 3, 4}；

という1次元配列の宣言ではint型で要素が4個の配列msz006を宣言するとともに，

　　msz006［0］ ＝ 1
　　msz006［1］ ＝ 2
　　msz006［2］ ＝ 3
　　msz006［3］ ＝ 4

という要素すべての初期化を意味します。

2次元配列の初期化も同様です。

　　float rx93［2］［3］＝ {
　　　　1.14， 1.73， 2.0，
　　　　2.23， 2.49， 2.64 }；

という宣言はfloat型で要素が6（＝2×3）個の配列rx93を宣言するとともに，

　　rx93［0］［0］ ＝ 1.14

intの省略

ところで，日常の話し言葉もそうですが，C言語においても頻繁に使用される単語は省略される傾向にあります。intがその例です。C言語では変数の宣言の中で，

short int	→short
long int	→long
unsigned int	→unsigned
unsigned short int	→unsigned short
unsigned long int	→unsigned long

という省略がよく行われます。また，

　　int　rx178；

などというなにも頭につかないint型の変数の宣言でもintを省略して，

　　rx178；

だけにしてしまうこともあります（たいていエラーになるが文法的には正しい)。しかし，この例は rx178 が果たして変数の宣言なのかどうかわかりにくいので使用するべきではないでしょう。他人の書いたプログラムを読むときの基礎知識として知っておいてください。

rx93 [0] [1] = 1.73
rx93 [0] [2] = 2.0
rx93 [1] [0] = 2.23
rx93 [1] [1] = 2.49
rx93 [1] [2] = 2.64

という初期化を意味します。順番に並べた値がどの要素の初期値になるかは，配列要素がメモリに並ぶ順番を意識していればわかるでしょう。また，先に2次元配列は1次元配列を要素とする1次元配列であるといいましたが，2次元配列の初期化ではこのことを強調するために，先の例を，

```
float rx93 [2] [3]= {
        {1.14, 1.73, 2.0 },
        {2.23, 2.49, 2.64} } ;
```

と記述することもできます。この表現では3個の要素を持つ1次元配列が2組並んでいるという状況がひと目でわかりますね。

式と演算子

C言語は豊富なデータ型を持っていますが，データとデータを演算するための演算子も非常にたくさんのものを取り揃えています。表2にC言語の演算子の一覧表を示します。この演算子はおおざっぱには次の4種類に分類できると思います。

●通常のプログラミング言語にもあるもの

+ − * / % <
<= > >= == != && || !

●プログラムをスマートに表現するもの

++ −− = ?:, += −=
*= /= %= ……

●アセンブリ言語レベルの処理を行うもの

& ^ | << >> [ ] −> . * &

表2　演算子の種類(上ほど優先強)

<table>
<tr><th>演算子</th><th>種類</th><th>結合規則</th></tr>
<tr><td>( )　[ ]　.　−></td><td>メモリ</td><td>左から右</td></tr>
<tr><td>−　‾　!　*　&
++　−−　sizeof　( )</td><td>単項</td><td>右から左</td></tr>
<tr><td>*　/　%</td><td>乗除</td><td>左から右</td></tr>
<tr><td>+　−</td><td>加減</td><td>左から右</td></tr>
<tr><td><<　>></td><td>シフト</td><td>左から右</td></tr>
<tr><td><　>　<=　>=</td><td>非等値</td><td>左から右</td></tr>
<tr><td>==　!=</td><td>等値</td><td>左から右</td></tr>
<tr><td>&</td><td>and</td><td>左から右</td></tr>
<tr><td>^</td><td>eor</td><td>左から右</td></tr>
<tr><td>|</td><td>or</td><td>左から右</td></tr>
<tr><td>&&</td><td>論理積</td><td>左から右</td></tr>
<tr><td>||</td><td>論理和</td><td>左から右</td></tr>
<tr><td>?　:</td><td>三項</td><td>右から左</td></tr>
<tr><td>=　*=　/=　%=
+=
−=　<<=　>>=
&=　|=　^=</td><td>代入</td><td>右から左</td></tr>
<tr><td>,</td><td>カンマ</td><td>左から右</td></tr>
</table>

●その他

(type)　sizeof

以下ではC言語を特徴づける後半の3種について解説します。

スマートに表現する

データ型の宣言ではintは省略される傾向にありますが，それに限らずC言語は省略や簡潔な表現を好んで採用しています。簡潔な表現は，ときにプログラムを暗号化してわけのわからないものにもしますが，少ない文字数で多くの情報を与えてくれるので，うまく使えばプログラムを読みやすく，そして理解しやすくしてくれます。

●++ −−

これらは変数の内容を単位数(通常は1)だけ増加（++）あるいは減少（−−）するための演算子です。プログラム中には，

n=n+1;

とか，

n=n−1;

といった表現がよく現れます。これらは，

n++　あるいは　++n

とか，

n−−　あるいは　−−n

と表現することができます。

ただこういう単純な置き換えのためだけならこういう演算子は必要ありません。ここで重要なのは++（あるいは−−）が変数のどちら（右か左）につくかということです。左についた場合，変数値の増減は評価の前に行われます。一方，右についたときは変数値の増減は評価のあとに行われます（評価とはその変数の値を使うこと）。

また++や−−は演算子というくらいですからそれを適用したものはなにかの値を返します。すなわち，++（あるいは−−）が変数の左につく場合は値を増減させる前の変数の値が返り，右につく場合は値を増減させたあとの変数の値が返ります。

n++

と，

++n

は単体で使う分には同じですが，式の一部として使われるときは事情が異なります。

(n++)+4

はn+4という値になりますが，

(++n)+4

はn+5という値になるのです。これはほかの言語なら，

n+4; n=n+1;

あるいは，

n=n+1; n+4;

と2つの式で書かれるべきものですが，C言語ではこれをまとめてひとつの式で表現できるようになっているのです。

また，nが配列の添字となったときも，

rx78 [n++]=1;

はrx78 [n] に1を代入することですが，

rx78 [++n]=1;

rx78 [n+1] に1を代入することです。代

新しい型を作る

Cでは既存のデータ型を組み合わせて新しいデータ型を作成(再定義)することができます。

たとえば，C言語のプログラムで多用されるint型は16ビットCPUでは16ビット長，32ビットCPUでは32ビット長ですから，両者のあいだでプログラムの直接的な互換性はありません。それらに互換性を持たせるためには16ビットCPU側ではintをlong intに書き換えることが必要になります。

しかし，これでは同じプログラムが2種類になってあとの修正や改良が大変です。もし，32ビットCPUではint型，16ビットではlong int型を意味するようなデータ型（natural型とでも名づけましょう）を作ることができれば，その型を使うことでプログラムをひとつですますことができます。

C言語ではtypedefという宣言で新しいデータ型を作り出すことができるのです。これは，

```
typedef　古いデータ型名
         新しいデータ型名；
```

という形式で使用します。なおtypedefとはtype（型）のdefinition（定義）に由来しています。

上の例の場合，16ビットCPUでは，

```
typedef long int natural ;
```

という宣言，32ビットCPUでは，

```
typedef int       natural ;
```

という宣言をプログラムの先頭につけ加えておき，プログラム中ではintの代わりにnaturalというデータ型を使えばいいことになります。たとえば，

```
natural zg, gzz ;
```

という宣言はnatural型（実体はint型またはlong int型）の変数zgとgzzの宣言を意味します。

この例はあまりに単純です。初めからlong int型に統一すればいいじゃないかといわれると返す言葉がありません(まったくそのとおりです)。

実際には，typedefはこのようなデータ型の単純な置き換えよりも，配列などのように複雑なデータ型を単純なデータ型に再定義するために使用されます。たとえば，3×3行列を用いた数値演算を行う場合，プログラム中には，

```
double g[3][3], z[3][3] ;
```

といった2次元配列（3×3行列）の宣言が多く現れます。このような場合，

```
typedef double matrix[3][3] ;
```

という宣言によってdouble型データを要素とする3×3行列を示すデータ型，すなわちmatrix型を定義すれば，上の宣言は，

```
matrix g, z;
```

で置き換えできてしまうのです。もちろん，gやzの要素はg[1][2]とかz[2][3]のように通常の配列と同じ方法で参照します。このように複雑なデータ型をtypedefで定義することにより，gやzが3×3行列であることを強調でき，また表現が簡潔になってプログラムが読みやすくなるのです。

入後はどちらもnの値は1増加しています。

●＝

これは代入を表します。FORTRANやPASCALなどの言語とは異なり，C言語ではこれは演算子として定義されています。そして＝は代入すべき値をそのまま値とします。このためC言語では，

a＝b＝c＝d＝100;

といった式が可能になります。これは，

a＝（b＝（c＝（d＝100)））;

という意味で，100という値をd，c，b，aの順に代入していきます。

●？：

これは，いわゆる条件式です。ある式を評価してその値が0かそうでないかによって返す値を選択するための演算子です。

(a＞100） ? a＋10 : a－10 ;

という式は（a＞100）という式が0でなければa＋10を0ならa－10を値とすることを意味します。一般には，

if (a＞100) n＝a＋10; else n＝a－10;

という条件文は条件式を用いて，

n＝ (a＞100) ? a＋10 : a－10;

とするほうがnに値を代入するという最終目的が明確なのでよいとされています。

●，

これはなかなか凄まじい演算子です。数個の式を評価して最後に評価した式（いちばん右側にある）の値をその値とします。

a＝a＋2, b＝c＋d, d＝3, e＝f＋10

という式は左から順にaに2を加え，cとdの内容を加えてbに代入し，dに3を代入し，fの内容と10を加えたものをeに代入したあと，最後に評価したf＋10という値をその演算子の値とします。上の場合，式を4つの文に分けて，

a＝a＋2; b＝c＋d; d＝3; e＝f＋10;

と記述すればいいと思われるかもしれません。それは，if文を，

if (a＝＝10) b＝12, a＝100;

と記述するよりも，

if (a＝＝10) {b＝12; a＝100;}

としたほうが素直なのと同様の理由です。しかし，C言語の文法では，たとえばfor文の初期設定，終了条件あるいは変数の増分を指定するためのフィールドなど，ただひとつの式しか記述を許していないところがあります。このフィールド内で2つ以上の動作をしようとするとき，これらの演算子は威力を発揮します。

●＋＝　－＝　＊＝　／＝　％＝　…

これらの演算子はただ式を省略するだけで深い意味はありません。

a ＝ a 演算子 b

という形式の式を，

a 演算子＝ b

と記述するだけです。

しかし，

a [i＋j] [b＊c＋4] [i＋2]
　＝ a[i＋j] [b＊c＋4] [i＋2]＋b [i] [j＋c]

などという複雑な式よりも，

a[i＋j] [b＊c＋4] [i＋2] ＋＝ b[i] [j＋c]

のほうが見やすいのは確かでしょう。

アセンブリ言語の処理を行う

C言語はアセンブリ言語を強く意識しています。このためアセンブリ言語でできることの多くを言語仕様に取り込もうとしているところがちらほらと見受けられます。その最たるものが演算子で，ビットごとの論理演算やシフトはほかの言語ではちょっとお目にかかることができません。

また，ポインタ関連の演算子はアセンブリ言語特有のアドレッシングをもろに言語仕様に取り込んだものです。

●&　|　^　¯

これらの演算子はビットのAND（&），OR（|），Exclusive OR（^），NOT（¯）を行うためのものです。通常のCPUが命令セットとして備えている論理演算をC言語では直接実行することができるのです。

●<<　>>

これらの演算子はシフト操作を行います。演算子の左側にくる被シフトデータがunsigned型（符号なし）であれば論理シフトが行われ，そうでなければ（符号つき）算術シフトが行われます。さすがにC言語ではローテイト操作はないみたいですが，もしあればシフト操作は完璧でしたね。

●[ ]　－＞　.　＊　&

これらはポインタ型を扱うための演算子です。詳しくは，あとでポインタ型と一緒に説明することにします。これらの演算子はCPUの持っている一般的なアドレッシングを実現するためのものと考えられ，

[ ]	→	インデックス
－＞.	→	ディスプレイスメント
＊	→	メモリ（レジスタ）間接
&	→	実行アドレス計算

と対応づけることができます。

その他の演算子

●（type）

C言語では＋や＊などの2項演算子で異なるデータ型同士の演算を許しています。このとき演算はより弱いデータ型を強いほうのデータ型に変換することで行います。データ型の異なる演算では自動的に型変換を行ってくれるのですが，この型変換を明示的に指定することができます。それがキャスト（鋳造するの意）です。キャストは変換すべき目的のデータ型を（と）で囲って変数や定数につけることで行います。

(unsigned int) msz010

という表現はmsz010をunsigned int型に変換することを意味します。プログラムを読む側としては異なるデータ型間の演算は弱いデータ型を強いデータ型にキャストしておくほうが読みやすいと思います。

キャストを忘れてならないのは異なる大きさのデータ型を示すポインタへの型変換ですが，これはあとに譲りましょう。

●sizeof

これはデータ型の大きさ（バイト数）を返す演算子です。XCでは，

sizeof (long int)	→	4
sizeof (short int)	→	2
sizeof (char)	→	1
sizeof (float)	→	4
sizeof (double)	→	8

が値として返ってきます。

sizeof演算子はあまり見かけることはないのですが，typedefで宣言されていて，プログラマには大きさがわかってない配列の大きさなどを知るときに使用されます。

いくつかのint型データを要素とする1次元配列dim1が定義されているとしましょう。さらに変数xとyが，

dim1 x, y;

という宣言で定義されているとき，変数xをyに代入するためのプログラムは，

for (i＝0;i<sizeof (dim1)/sizeof
　(int) ;i＋＋)
　y [i] ＝ x [i] ;

と書くことができます。

関数の世界

C言語のプログラムをひと言でいうと，変数の定義と関数の定義が並んだものということになります。このとき，ひとつのプログラムには必ずひとつのmainという名前の関数がなければなりません。C言語のプログラムの実行とはこのmain関数を実行することにほかならないからです。

main関数はその中で別の関数を呼び，また呼ばれた関数は別の関数を呼びます。ある関数での処理が終われば，その関数はな

んらかの値を返し，値を受け取った側（これも関数）はその値を基に別の処理をします。そして，最終的に制御はmainに戻って無事プログラムの終了となるわけです。

多くの場合，関数で行われる処理は宣言された変数を使い，その値を変更することです。変更された変数の値に従って関数はいろいろな動作をするのです。図4にC言語のプログラム実行の概念図を示します。

結局，C言語では関数しか実行しませんから関数を定義することがプログラミングになります。関数定義は次の形式です。

```
戻り値データ型
    関数名（引数，引数，……）
        引数データ型宣言
        複合文（関数の本体）
```

関数の戻り値は表1に示した10種類のデータ型と，後述のポインタ型，構造体，共用体のどれか（配列はだめ）あるいはvoid型ということを押さえておきましょう。

typedefで定義したデータ型もそれらのデータ型と実体が同じであれば戻り値にすることができます。void型は値を返さない関数(本体にreturn文がないか，return文があっても値を返していない関数であるが，現実には不定値を返している）を明示するために使用します。voidはCコンパイラに対する指示で，void型の関数の戻り値を式の一部として使用したときにエラーメッセージを出させるためのものです。

また，int省略の法則(？)に基づいて，関数の戻り値がint型の場合は戻り値データ型の宣言が省略されたり，引数のデータ型宣言でもint型の引数の宣言は省略されることがあります。たとえば，関数fが，

```
f (x, y, z)
    char x;
  { …… }
```

のように定義されているとき，関数の戻り値はint型（宣言が省略されている）で，引数の型はxがchar型，yとzがint型（宣言が省略されている）です。ただし，intが省略できるからといって先の例を，

```
f (x, y, z)
    char x;y, z;
  { …… }
```

と記述することはできないようです（文法的には正しいはずだが)。まあ，引数の宣言に関しては，intを省略してもそれほどメリットがあるとは思えませんし，プログラムを読みにくくするだけですから，皆さんは真似をしないようにしましょう。

関数呼び出しとコンパイラの都合

C言語のプログラムは関数定義の並びです。プログラムを実行するためには定義された関数を呼び出さなければなりません。すると，呼ばれた関数はなんらかの値を返してきます。一般にこの関数からの戻り値は式の一部として利用されます。

実はこのとき，ある注意をしないとまったく予定外の結果を出してくることがあります。これはコンパイラの都合というものに関係してきます。

まず，XCでリスト1a)に示すプログラムをコンパイルして実行してみてください。ここで呼んでいるsin関数はXCのライブラリ関数のひとつで引数のサイン（正弦）を計算する関数です。sin関数の引数は$\pi/2$に近い値ですから，その戻り値は1に近い値になるはずです。ところが，実行結果にはとてつもなく大きな値がprintf関数で表示されます。これはどうしたことでしょうか。

答えは単純です。私たちはsin関数の結果がdouble型であることを知っているのですが，Cコンパイラはそう思っていないからです。なぜなら，プログラム中にsin関数の戻り値のデータ型はどこにも宣言されていません。このように関数の戻り値のデータ型が宣言されてない場合，Cコンパイラは戻り値をint型だと判断して処理を続けてしまうのです。そこで，sin関数の戻り値のデータ型を宣言したものがリスト1b)です。リスト1a)との違いはプログラムに，

```
double sin ( );
```

という1行を追加した点です。こういう，

```
データ型　関数名 ( );
```

という宣言は関数の戻り値のデータ型を明示するためだけに使用します。この宣言には引数も関数の本体もありません。リスト1b)をコンパイル，実行してください。今度はうまく1に近い値が表示されますね。

それではリスト2a)を見てみましょう。こちらではmain関数の定義と同じプログラム内でdouble型で与えられた2つの引数の和(double型)を戻り値とする関数addを定義してあります。main関数内の呼び出し側では2つの$\pi/2$に近い値をadd関数への引数としていますから，その戻り値はπに近い値が表示されるはずです。さて，コンパイル結果はどうでしょう。きっとエラーが出てコンパイルできなかったと思います。これもリスト1a)と同じ理由です。

このエラーをなくすためにはadd関数の呼び出しよりも前の行に，

```
double add ( );
```

という宣言を入れてadd関数の戻り値はdouble型だとCコンパイラに知らせておけばいいのです。これがリスト2b)です。

ある関数がライブラリになっているとか別のファイルで定義されているなら戻り値のデータ型がわからないといわれてもしかたありませんが，同じファイルで定義している関数の戻り値を前もって教えてやらねば正しく処理されないというのはCコンパイラの都合という以外ありませんね。余談ですがX-BASICは関数を使う位置より後ろで定義されていても正しく処理します。

ところで，add関数の定義をリスト2c)のようにmainプログラム（呼び出し位置）より前の行に持ってくると，もはや，

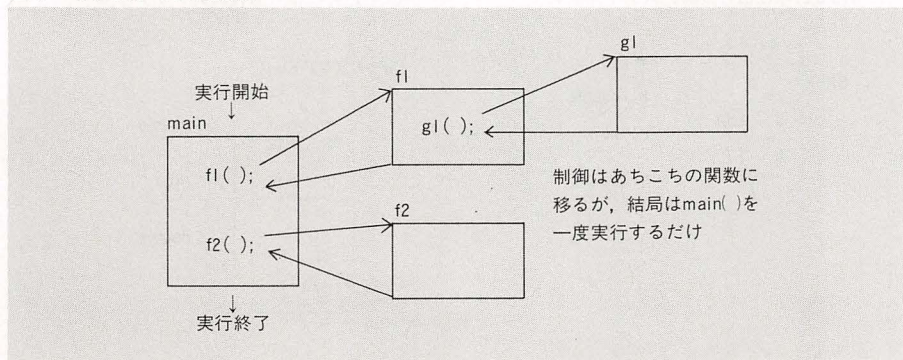
```
double add ( );
```

の行は不要になります。さすがに呼び出し

リスト1

a)
```
 1: /*
 2:         Cコンパイラの都合（その1）
 3: */
 4: double pi2=1.578;   /* π/2 */
 5: double x;
 6: main()
 7: {
 8:     x=sin(pi2);
 9:     printf("sin(π/2)=%f¥n",x);
10: }
```

b)
```
 1: /*
 2:         Cコンパイラの都合（その2）
 3: */
 4: double sin();
 5: double pi2=1.578;  /* π/2 */
 6: double x;
 7: main()
 8: {
 9:    x=sin(pi2);
10:    printf("sin(π/2)=%f¥n",x);
11: }
```

図4　Cプログラムの流れ

よりも前に関数の定義があるとその戻り値のデータ型はわかってしまうからです。

引数の受け渡し

C言語は関数を主体とする言語でありながら引数の取り扱いはかなりいい加減です。現状では，関数の呼び側と呼ばれ側で引数の型や引数の個数が合っているかどうかのチェックは行われません。

これは上手に使えば便利な場合もありますが，初心者にとってはもっとも誤りを生じやすい部分でしょう。なにしろ引数がでたらめでもコンパイルエラーにならないのですから，デバッグをするときの苦労は並大抵ではありません。しかし，このチェックのいい加減さがC言語の柔軟性を生み出していることも否定できません。

たとえば，引数の個数のチェックをしっかり行うFORTRANやPASCALなどの言語ではprintf関数みたいに引数の個数が可変であるような関数を記述することはできません（必要なときはアセンブリ言語を使っていた)。ここでは引数のチェックをしないことがバグを生みやすいという点を認識したうえで，引数のチェックを行わないことを積極的に利用したプログラムを紹介しておきましょう。

XCを始め，多くのC言語では関数への引数はスタックを介して関数に渡されます。まずこのことを知識として知っておかなければなりません。図5に関数が引数つきで呼ばれた時点でのスタックの状況と，関数側での引数の参照のしかたを示しておきます（1988年8月号「Cとアセンブリ言語をリンクして使う」参照)。

リスト3は引数の型の一致をチェックしないことを利用したプログラムです。separateという関数を呼ぶときはdouble型の浮動小数点データを引数とします。この浮動小数点データ（64ビット）はスタックに下位32ビット，上位32ビットの順で積まれます。一方，separate関数自身はint型引数が2つである，すなわち2つの32ビット整数がスタックに積まれているものと思っていますから，double型浮動小数点データの上位32ビットを第1引数，下位32ビットを第2引数として処理するのです。要するに呼ばれた側でスタックにある引数の内容を勝手に読み換えて処理するわけです。

リスト3は浮動小数点データのビットイメージをそのまま整数型変数に代入するという高等（?）テクニックです。

リスト4は引数の個数をチェックしないことを利用したプログラムです。このプログラムで，maxという関数はスタックには十分なだけの引数が積まれているものとして処理を行っています。そのときの頼りは最初の引数で与えられるデータ個数で max関数はこの個数を信じて処理をします。ですから，このデータの個数より実際のデータが多い場合は余分なデータを無視してしまいますし，実際のデータの個数が少ない場合は足りないデータとしてスタックに積まれているゴミを処理してしまいます。

したがって，指定したデータの個数が実際のデータよりも多い場合はどういう値が返ってくるか予測できません（リスト4で最後に呼び出すmax関数の値はどうなったでしょうか)。データの個数を間違えないようにするのはプログラマの責任です。

ところで，引数のチェックを正しく行うため，ANSI規格ではプロトタイプ宣言が導入されています。これは引数のデータ型と個数を陽に宣言することで引数の誤用からくる間違いを最小限にする工夫です。

まあ，引数の個数についてはANSIのプロトタイプ宣言でも可変長な関数が書けるように姑息な救済法が用意されていますが，癖のあるC言語が一般受けするように変更されて普通の言語になっていくのを見るのはちょっと寂しい気がしますね。

局所変数と有効範囲

これまで述べてきた変数は原則的にはすべての関数から見える大域的変数でした。しかし，関数内で宣言する変数はその関数内でしか有効でない局所変数です。局所変数は原則的には自動的（automatic）です。

つまり，関数に入った時点で領域が確保され，関数から出るときにその領域は捨てられてしまいます。しかし，この表現は正確ではありません。正確には，局所変数はブロック構造を表す複合文の先頭で宣言する（先頭以外では宣言できない）変数で，その領域はブロックに入るときに生成されブロックを抜けるときに捨てられるのです。複合文とはいくつかの文を｛と｝で囲った文のことで，｛と｝で囲まれた部分をブロックといいます。関数宣言の本体部分はこの複合文の形式をしていましたね。

たとえば，

{int rx78; 文; 文; 文; ……}

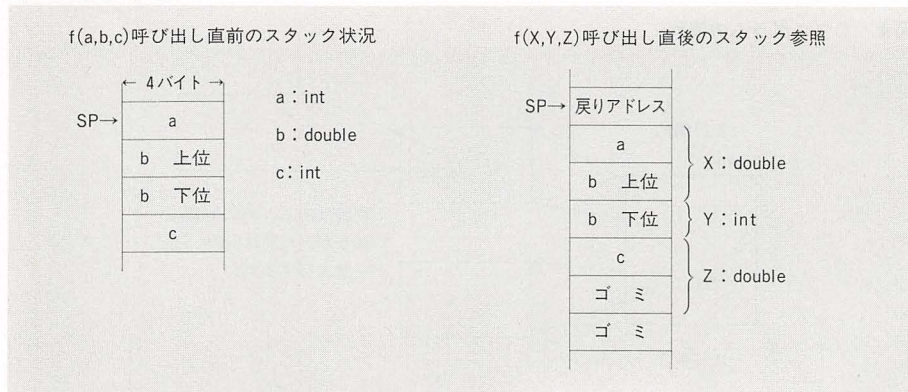
という複合文（ブロック）ではint型のrx78という変数を局所変数として定義していま

リスト2

```
a)  1: /*
    2:      Cコンパイラの都合（その3）
    3: */
    4: double pi2=1.578;   /* π/2 */
    5: double x;
    6: main()
    7: {
    8:   x=add(pi2,pi2);
    9:   printf("(π/2)+(π/2)=%f¥n",x);
   10: }
   11:
   12: double add(x,y)
   13: double x,y;
   14: {
   15:   return(x+y);
   16: }

b)  1: /*
    2:      Cコンパイラの都合（その4）
    3: */
    4: double add();
    5: double pi2=1.578;   /* π/2 */
    6: double x;
    7: main()
    8: {
    9:   x=add(pi2,pi2);
   10:   printf("(π/2)+(π/2)=%f¥n",x);
   11: }
   12:
   13: double add(x,y)
   14: double x,y;
   15: {
   16:    return(x+y);
   17: }

c)  1: /*
    2:      Cコンパイラの都合（その5）
    3: */
    4: double add(x,y)
    5: double x,y;
    6: {
    7:   return(x+y);
    8: }
    9:
   10: double pi2=1.578;   /* π/2 */
   11: double x;
   12:
   13: main()
   14: {
   15:   x=add(pi2,pi2);
   16:   printf("(π/2)+(π/2)=%f¥n",x);
   17: }
```

図5 引数の状態

す。rx78という変数の有効範囲はブロックの左端の{ から右端の }までです。ブロックを抜けると値そのものが消滅してしまうので，このrx78という変数はブロックの外側からは参照することができません。

関数の先頭で宣言する局所変数はともかく，関数本体とは別ブロックの局所変数は,

```
    if (x>y)    { int tmp;
         tmp=x; x=y; y=tmp;
    }
```

というふうに使用します。この例は変数xの値が変数yの値より大きい場合(ともにint型としておこう), xとyの値を入れ替えるという記述です。変数の値を入れ替えるためには片方の値を退避しておく一時的な変数がほしくなります。しかし，この変数は本当に一時的なものなので，関数の先頭でほかの局所変数と同じレベルで宣言するのは気が引けます(この部分で一度参照されるだけかもしれませんし)。ブロック内の局所変数はこんなときに有効です。

ブロック内での局所変数を上手に使えば，関数の先頭での局所変数の宣言を本当に重要なものだけに絞り込むことができて，プログラムを読みやすくできるのです。

ところで，ブロックの外部で宣言された変数とブロックの内部で宣言された局所変数が同じ名前のときもあります。このときは局所変数のほうが有効になります。もし，ブロックが入れ子になって,

```
    {
        int rx78, msz010;
        rx78=2;
          {
            int rx78;
            rx78=1;
            msz010=10;
          }
          rx78=rx78+3;
    }
```

という記述があったとすれば，内側のブロック({と}のあいだ)で宣言されたrx78は外側のブロックで宣言されているrx78とは無関係です。つまり外側のブロックでrx78に2が代入され，内側のブロックでrx78に1が代入されても，外側の,

rx78+3

という式の計算では内側のブロックのrx78 (=1)の値が使われず，外側のブロックでのrx78 (=2) の値が使われるのです。

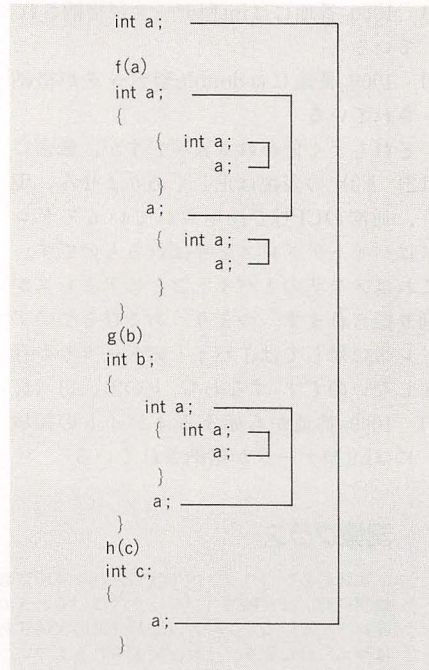
また，内側のブロックではmsz010という変数に10という値を代入していますが，この変数は内側のブロックでは宣言されていません。この場合はひとつ外側のブロックが参照されます。いまの例では外側のブロックで局所変数としてmsz010が宣言されていますからそれに10が代入されるのです。

それでは，もし外側のブロックでもmsz010が宣言されていなかったらどうでしょう。そのときはさらに外側のブロックが探されます。どこにもその変数が宣言されていなければ，最後は関数の外部の大域変数にたどり着きます。そして，大域変数にも宣言されていない場合はエラーです。

ポインタの秘密

ポインタはC言語でもっとも特徴的なデータ型です。PASCALにもポインタがありましたが，それは非常に難しい概念でした。

図6　局所変数の対応

```
int a;
f(a)
int a;
{
   { int a;
        a;
   }
   a;
   { int a;
        a;
   }
}
g(b)
int b;
{
    int a;
     { int a;
          a;
     }
      a;
}
h(c)
int c;
{
        a;
}
```

リスト3

```
 1: /*
 2:        引数の型の一致をチェックしない
 3:        ことを利用したプログラム
 4: */
 5: /**********************************
 6:  *   引数で与えられた倍精度浮動小数点データの
 7:  *   符号、指数部、仮数部を取り出す関数
 8:  *
 9:  *   separate(浮動小数点データ)
10:  * sign  ←  符号
11:  * exp   ←  指数部
12:  * man0  ←  仮数部(上位)
13:  * man1  ←  仮数部(下位)
14:  **********************************
15:  */
16:
17: int sign,exp,man0,man1;
18: double x;
19: void separate(high,low) /*2つの整数で受ける*/
20: int high,low;
21: {
22:     sign = (high<0) ? 1 : 0;
23:     exp  = (high>>20)&0x7ff;
24:     man0 = (high&0xfffff)|0x100000;
25:     man1 = low;
26: }
27:
28: main()
29: {
30:   separate(x = 1.2);
31:   printf("%9.6f S=%d EXP=%03x MAN=%08x%08x¥n",
32:     x, sign, exp, man0, man1       );
33:   separate(x = -2.4);
34:   printf("%9.6f S=%d EXP=%03x MAN=%08x%08x¥n",
35:     x, sign, exp, man0, man1       );
36:   separate(x = 3.1415);
37:   printf("%9.6f S=%d EXP=%03x MAN=%08x%08x¥n",
38:     x, sign, exp, man0, man1       );
39: }
```

リスト4

```
 1: /*
 2:        引数の個数をチェックしない
 3:        ことを利用したプログラム
 4: */
 5: /*********************************
 6:  *   最大3個までの整数の
 7:  *   最大値を値とする関数
 8:  *
 9:  *   max(個数,データ,データ,...)
10:  *********************************
11:  */
12:
13: int max(n,x0,x1,x2)
14: int n,x0,x1,x2;
15: {
16:     int x;
17:     x=x0;
18:     if(n<2) return(x);
19:     x=(x1>x)? x1 : x;
20:     if(n<3) return(x);
21:     x=(x2>x)? x2 : x;
22:     return(x);
23: }
24:
25: main()
26: {
27: /* 普通の使い方 */
28:
29:   printf("      max(1,5)=%d¥n", max(1,5)      );
30:   printf("    max(2,8,2)=%d¥n", max(2,8,2)    );
31:   printf("  max(3,3,5,4)=%d¥n", max(3,3,5,4)  );
32:
33: /* 指定したよりも引数の個数が多い場合 */
34:
35:   printf("max(2,3,2,4,1)=%d¥n",max(2,3,2,4,8));
36:
37: /* 指定したよりも引数の個数が少ない場合 */
38:
39:   printf("    max(3,1,2)=%d¥n",max(3,1,2)     );
40: }
```

C言語のポインタは単純明快です。ポインタとはデータをポイントする（指し示す）ものという意味です。ここでいうデータとは変数や配列のことです。何度も述べてきたように変数や配列はメモリ（ときにはレジスタ）上にその実体（領域）があります。ですから，ポインタとはメモリのアドレス（番地）を保持している変数にすぎません。次のような表現を考えましょう。

1) 1000_H番地にはchar型データが格納されている
2) 1000_H番地にはint型データが格納されている
3) 1000_H番地にはdouble型データが格納されている

どれもよく使われる表現ですが，厳密には2）と3）の表現は正しくありません。現在，通常のCPUで採用されているアドレスはバイトアドレスと呼ばれるものです。これはメモリの1バイトごとにアドレスが割り振られます。つまり，あるひとつのアドレスに対しては1バイトのデータしか存在しないのです。すなわち，上の2)，3）は，

2) 1000_H番地から始まる4バイトの領域にはint型データが格納されている
3) 1000_H番地から始まる8バイトの領域にはdouble型データが格納されている

が正しい表現です。

いま，ポインタ変数に1000_Hという値が入っていたとしましょう。このとき，1000_Hというのはメモリ上の1点（1バイト）を示すアドレスですが，その1000_H番地にどんなデータ型が格納されているかで意味が微妙に違ってきます。同じ1000_H番地でも，

1) char型を格納した1000_H番地
2) int型を格納した1000_H番地
3) double型を格納した1000_H番地

というように異なる意味を持っています。

もし，ポインタ変数（1000_H番地）の指し示す内容をくださいといわれたとき，1000_H番地にある1バイトのデータだけを渡すのは少々早とちりです。ポインタ変数がchar型のデータを指し示す場合はそれでいいのですが，int型を指し示す場合は1000_H～1003_H番地の4バイトデータを渡さなければなりませんし，double型を指し示す場合は1000_H～1007_H番地から取り出した8バイトデータを渡さねばなりません。

逆に，ポインタ変数の指し示す場所にデータを書く場合も同様にデータ型のバイト数を考慮しなければなりません。

この例からわかるようにポインタ変数はメモリ上の1点を指し示すアドレスを保持するだけですが，それが指し示すデータ型を与えてやらなければ使いものになりません。そのため，C言語でポインタ変数を宣言するときはそのデータ型とともに宣言します。ポインタ変数の宣言は通常の変数の宣言と同じ形式で行います。ただ違うのは変数名の前に＊をつけるということだけです。たとえば，

```
int     *rx78;
double  *msz006;
```

という宣言はint型を指し示すポインタ変数rx78と，double型を指し示すポインタ変数msz006を宣言することを意味します。

ポインタ変数が指し示すものを参照するためには＊という演算子を用います（掛け算と同じ記号ですが使い方が違う)。上のような宣言がなされているとき，

```
a = *rx78 + 100 ;
```

はrx78の指すもの(int型データ)と100を加えてaに代入するという意味です。また，

```
*msz006 = 1.234 ;
```

という式はmsz006の指し示す場所に1.234

記憶クラス

関数およびブロック内で宣言される局所変数は原則的には自動的（ブロック内にいるときのみ存在する）です。一方，関数外部の大域変数は与えられた値を，それが変えられるまでいつまでも保持しています。このような変数を静的(static)であるといいます。C言語ではすべての変数はこの自動的と静的の2つに分類することができます。この区別を記憶クラスといいます。要するに，変数はある瞬間にだけ存在する（自動的）か，永久に存在する（静的）かのどちらかなのです。

アセンブリ言語的なレベルでいえば，静的な変数はメモリ上にラベル（変数名）つきで領域が確保されるのに対して自動的な変数はスタック（またはレジスタ）上に領域が確保されるのです。

変数の初期化について考えてみましょう。静的な変数はあらかじめ領域が確保されていてそこに初期値が格納されています。これはプログラム実行前（コンパイル時）での初期化です。自動変数は宣言された時点で毎回領域が確保されるのでそのときに初期化されます。これはプログラム実行中の初期化です。静的な変数の初期化はコンパイル時に行われますからその初期値は定数値でなければなりません。しかし，自動的な変数については定数値によっても別の変数の値によっても初期化することができます。これが実行時に初期化を行うメリットですね。しかしながら，自動的変数の初期化は単に文の記述を省略するという意味しか持っていません。たとえば，

```
{int a=10;………}
```

という自動的変数の宣言時の初期化は，

```
{int a; a=10;………}
```

という表現の省略形にすぎません。単に簡潔さの違いだけです。

興味深いのは自動的な配列の初期化です。

```
{int x[3]={1,2,3};………}
```

という記述は，

```
{int x[3];x[0]=1: x[1]=2; x[2]=3;…}
```

と同じ意味ですが簡潔さは格段に違いますね。ただ残念なことに，かのK&RやANSI規格では認められているこの自動的な配列の初期化はXCでは許されていません（X-BASICではできるのに！)。

局所変数は自動的だといいましたが，局所変数を明示的に静的な変数として宣言することも可能で，その際は従来の変数宣言に対してstatic（静的な)というキーワードをつけます。たとえば，

```
ransu( )
{
    static int r = 1729;
    return(r *= 131071);
}
```

という疑似乱数を発生させる関数ransuの定義では変数rを静的なint型の局所変数として宣言してあります。変数rは局所的変数でありながらstaticと宣言されたためにコンパイル時に領域が確保されます。この領域は関数を抜けても消滅することなくいつまでも存在するのです。いまその領域には1729という初期値が与えられています。ransu関数は呼ばれるたびに変数rの内容を書き換え，その値を関数の戻り値としますが，その書き換えた値は次にransu関数が呼ばれたときに使用します。このように変数rの前の値を使うということは，その変数が静的だからできる芸当です。もし，上のransu関数で変数rが自動的なら常に同じ値（1729＊131071）が戻り値として返ってきて乱数としての意味をなしません。

静的な変数を宣言するためにstaticというキーワードがありますが，逆に自動的であることを明示するための宣言もあります。それがauto(automaticの略）です。autoもstaticと同様に変数宣言の前につけて，

```
auto int a;
```

などという形式で使用します。しかし，大域変数を自動的と宣言することはできませんし（意味がない)，局所変数もなにもいわなければ自動的とみなされるのでautoというキーワードを使うことはまずないでしょう。

自動変数は原則的にはスタック上に領域が確保されます。しかし，registerというキーワードをつけることによって，自動変数をCPUのレジスタに割り当てる指示も可能です。これをレジスタ変数といいます。このレジスタ変数の宣言はautoやstaticの宣言と同様です。たとえば，int型の自動的変数aをレジスタ変数としたい場合は，

```
register int a;
```

と宣言します。変数がレジスタにあればそれを参照するための時間はスタック（メモリ）よりも格段に速いので，頻繁に使用するfor文などでの制御変数はレジスタ変数に宣言すると高速な処理が期待できます。

しかし，実際に自動変数がレジスタへ割り当てられるかどうかはCコンパイラの都合で決まります。いくらプログラムでregister指定しても適当なレジスタがない場合は割り当てることができません（たとえば8086ではレジスタ変数用にsi，diという2つのレジスタしか用意していない)。この場合はregister指定は無視され，通常の自動変数として扱われます。registerという指定はその変数を「なるべくレジスタに割り当ててください」という意味でしかないのです。もっとも，最近のCコンパイラは自動変数をなるべくレジスタに割り当てようとするのでregisterという指定自体が意味をなさなくなってきているのも確かです。

という値を格納するという意味です。

ところで，ポインタ変数の宣言で使われている＊という記号はポインタ変数の指し示すものを取り出す演算子と同じものと思ってかまいません。このとき，先の宣言は「rx78の指し示すものはint型ですよ」とか「msz006の指し示すものはdouble型ですよ」と読み換えることができます。

さて，ポインタ変数はアドレスを保持する変数ですが宣言しただけでは値を持っていません(ゴミが入っている)。おそらくポインタ変数を宣言した直後にいきなり，

　　＊rx78 ＝ 100;

などとするとバスエラーが起きてしまうでしょう。このため，実際に使用するためには前もって値を与えてやらねばなりません。そのためにC言語では変数のアドレスを取り出す＆演算子が用意されています（ビットのANDと同じ記号ですが別物です)。

変数名の前に＆をつけるとその変数のアドレスを取り出すことができます。当然，register宣言した変数（アドレスがない）に＆をつけることはできません。rx78がポインタ変数，xがある変数だとすれば，

　　rx78 ＝ &x ;

という式によって，ポインタ変数rx78にxという変数のアドレスを与えることができます。このときはポインタ変数に値を与えるのであって，その指し示す場所に値を与えているのではないので＊はついていません。なお，＊&xという式は，xという変数のアドレスが指し示すものという意味ですから，xそのものになります。

ところで，アドレスというものは（符号なしの）整数値と同一視できます。したがって，ポインタ変数にアドレス値を絶対的な整数値で与えたければ，

　　rx78 ＝ 0x10000 ;

などという方法も考えられます。しかし，この場合10000_H番地がプログラムで使っていい領域かどうかわからないので（もしかしたらOSのワークエリアかもしれない)，こんな危険な真似はしてはいけません。

このほかにポインタ変数に値を与える方法としては別のポインタ変数の値を直接代入することも考えられます。○○型へのポインタ変数といってもその値はひとつのアドレス値でしかないわけですからポインタ相互間の代入はできて当たり前でしょう。

ところで，変数のアドレスを取り出す場合の特殊な場合について述べておきます。まず，”と”で囲まれた文字列はそれ自体がアドレス値です。C言語では文字列とは要素が文字列の各1文字であり，最後の要素がNULL (0) であるchar型配列として扱われます。すなわち，

　　”Z gundam”

という文字列は，

```
char str [ ]= {
    'Z', ' ', 'g', 'u', 'n', 'd', 'a', 'm', 0
} ;
```

というchar型配列と同じものです。後者はstrという名前を持っていますが，前者には名前がなく”Z gundam”で直接配列の先頭アドレスを示します。よって文字列はポインタ変数にそのまま代入できるのです。ただ，文字列の要素はchar型ですからそれを異なるデータ型（たとえばintやdouble）のポインタ変数に代入すると，もはや元の文字列とは違った意味になってしまいます。

次は配列です。配列名は直接アドレス値として使用することができます。配列名自体がメモリに並んだデータの先頭アドレスを指し示すものですから当然ですね。あるint型の1次元配列aでは，aと&a[0]は同じ意味になります。

さて，ここらでポインタ変数の効能について考えてみましょう。これはズバリ，関数から複数の戻り値を受け取るのに役立ちます。一般に関数というものは戻り値をひとつしか持ちません。しかし，たとえば，マウスカーソルの座標を返す関数がほしくなったとします。このとき戻り値はX座標とY座標の2つが必要です。そこで，引数として変数のアドレス値を渡すのです。そのアドレス値は戻り値を格納しようと思っている変数のアドレスです。

たとえば，マウスカーソルの座標を得るためには，戻り値を格納するための変数x，yを宣言しておいて，

　　mspos (&x, &y) ;

というぐあいにそのアドレスをマウスカーソルの座標を求める関数msposに渡せばいいのです。mspos関数の側では，

```
mspos (x, y)
int  *x, *y;
 {……}
```

と宣言して値を返すべき変数のアドレスをポインタ変数として引き取ります。この関数内ではカーソル座標を計算したのち，それぞれのポインタ変数の指し示す位置に，

　　＊x ＝ マウスカーソルのX座標；
　　＊y ＝ マウスカーソルのY座標；

として座標の値を入れればいいのです。リスト5に変数のアドレスを関数に渡して複数の値を引き取る例を示します。

ポインタの演算

ポインタに対して許されている演算は，その指し示す内容を参照する以外では，値の加算と減算です。ただし，この加減算は通常の場合と少し異なっています。ポインタとはなにか（データ）を指し示すものですから，たとえば，それを1増加するということは次にくるなにかを指し示すことになります。逆に1減少させることはひとつ前のなにかを指し示すことになります。

ポインタ変数に±1を加えるとき，

char型へのポインタなら	±1
short型へのポインタなら	±2
int型へのポインタなら	±4
float型へのポインタなら	±4
double型へのポインタなら	±8

だけポインタ変数の保持しているアドレス値が変化します。これはポインタが指し示

リスト5

```
 1: /*
 2:      2つ以上の値を返す関数の例
 3: */
 4: #include <time.h>
 5: /*
 6:    現在の時間の時間と分と秒を返す関数
 7:
 8:    jikan(hour,min,sec);
 9:        int *hour -- 時間が入るアドレス
10:        int *min  -- 分が入るアドレス
11:        int *sec  -- 秒が入るアドレス
12: */
13: void jikan(hour,min,sec)
14: int *hour,*min,*sec;
15: {
16:     int  systime;
17:     char *moji;
18:
19:     time(&systime);          /* time でシステム時間を求めて */
20:     moji=ctime(&systime); /* ctime で文字列に変換する    */
21:
22:     *hour=(moji[11]-'0')*10+(moji[12]-'0');/*12,13番目*/
23:     *min =(moji[14]-'0')*10+(moji[15]-'0');/*15,16番目*/
24:     *sec =(moji[17]-'0')*10+(moji[18]-'0');/*17,18番目*/
25: }
26:
27: main()
28: {
29:     int  ji,bun,byo;
30:
31:     jikan(&ji,&bun,&byo);/*ji bun byo に値を入れてね*/
32:
33:     printf("現在 %2d 時 %2d 分 %2d 秒です¥n",ji,bun,byo);
34: }
```

すデータ型の大きさ（バイト数）と等しくなっています。

もっと具体的に説明しましょう。いま,

int rx78 [5]= {1,2,3,4,5} ;

int *p = &rx78 [2] ;

という宣言があると仮定します。int型へのポインタ変数pは配列rx78の3番目の要素の位置に初期化されています。このときpを用いた式と配列のアドレスとの関係は,

p−2 → &rx78 [0]
p−1 → &rx78 [1]
p → &rx78 [2]
p+1 → &rx78 [3]
p+2 → &rx78 [4]

となります。つまり，pを順に変化させることは，その指し示す先であるint型の配列要素を順にたどることに等しいのです。

当然，ポインタ変数が指し示す先も,

* (p−2) → rx78 [0]
* (p−1) → rx78 [1]
* p → rx78 [2]
* (p+1) → rx78 [3]
* (p+2) → rx78 [4]

という関係になっています。

このようにポインタ変数はメモリ上に連続して並んだ同じデータ型のデータに対し，ある位置から何番目という番号を指定してその要素を参照するのに役立ちます。

早い話，それは配列と同じものです。実は，配列の要素を参照するときに添字を記述する［ ］は演算子なのです。rx78 [2]は，アドレスrx78を基準にして2番目の要素（添字は0から始まるので実際は3番目）を求めているとも考えられます。

実際，［ ］という演算子はポインタ変数にも適用できます。ポインタ変数pがあるときp [3] は *(p+3) とまったく同じ意味になります。逆にrx78が（1次元）配列のときrx78 [3] と *(rx78+3) は同じ意味です。なんとなくポインタ変数と配列名との関係がわかってきたでしょうか。

ここで［ ］という演算子について興味深い例をお目にかけましょう。先に，文字列とはchar型文字列を示すアドレス値といいましたが，これに［ ］演算子をつけることもできます。このとき，文字列の先頭から数えて指定した位置にある文字（char型データ）を取り出すことができます。

"0123456789abcdef" [i]

という表現は変数 i の 0 ～15という値に従って'0'～'9'，あるいは'a'～'f'という文字になります。つまり，［ ］演算子を使うと整数値を簡単に16進文字に変換できるのです。このような芸当ができる言語は私の知る限りC言語以外にありません。

ポインタ変数と配列名とはただ1点の違いを除いて同じものです。ポインタ変数にはアドレス値を格納する領域，つまり実体があります。しかし，配列名ではメモリの○○番地とコンパイル時に決められたアドレス値があるだけで実体はありません（その値はコンパイラだけが知っている)。したがって，ポインタ変数は代入などにより値を変更することができます。しかし，配列名に対応するアドレス値を変更することは不可能です（変更すべき実体がない)。

そう，ポインタ変数の特徴は配列とほとんど同じ能力を持ちながら，その値を変更できるということです。ポインタ変数には代入および加減の演算が許されますから，=，++，−−，+=，−=といった演算子を使ってポインタ変数の指し示す位置をプログラマに都合のいいように変更できるのです。ポインタ変数を変更しながら配列要素を参照するプログラム例として，リスト6に2つの文字列(char型の1次元配列)を参照するプログラムを示しておきます。

最後にポインタ変数へのキャストについて説明しておきましょう。あるデータ型を指し示すポインタ変数を別のデータ型を指し示すポインタ変数とみなしてその指し示すデータを参照したいときがあります。

たとえば，double型へのポインタ変数をint型へのポインタ変数とみなして内容を参照することを考えます。このときはdouble型のポインタ変数をint型のポインタ変数にキャストして内容を参照すればいいのです。前にも書きましたが，キャストとは目的となるデータ型を（と）で囲ったものです。変数の宣言に*をつけたのがポイン

リスト6

```
 1: /*
 2:     ポインタ変数を活用する例
 3: */
 4: /*
 5:      ある文字列の中に与えた文字列があるかどうかを
 6:      調べる関数（あればその位置を返す、なければ  -1）
 7:
 8:      instr(astr,pstr)
 9:        char *astr -- ある文字列
10:        char *pstr -- 与えた文字列
11: */
12: instr(astr,pstr)
13: char *astr,*pstr;
14: {
15:     char *v=astr;
16:     while(*v){
17:         /* 最初の文字が一致していれば続く文字が一致するか調べる */
18:        if(*v++ == *pstr){
19:             char *x=v;
20:             char *y=pstr+1;
21:             while(*y && (*x++ == *y++));
22:                         /* yが尽きるか不一致があるまで */
23:             if(*y==0)   /* yが尽きていたのなら見つけた */
24:                 return(v-astr-1);/* ポインタの差が位置 */
25:        }
26:     }
27:     return(-1); /* vが尽きてしまったら見つからなかった */
28: }
29:
30: main()
31: {
32:  char *pat="here is match";
33:  char *st1="here is not match, here is match!!";
34:  char *st2="here is";
35:  char *st3="here is here is match here is match";
36:
37:  printf("[%s] / %s =%d¥n",st1,pat,instr(st1,pat));
38:  printf("[%s] / %s =%d¥n",st2,pat,instr(st2,pat));
39:  printf("[%s] / %s =%d¥n",st3,pat,instr(st3,pat));
40: }
```

リスト7

```
 1: /*
 2:      ポインタ変数をキャストする例
 3: */
 4: /***********************************************
 5:  *   引数で与えられた倍精度浮動小数点データの
 6:  *   符号、指数部、仮数部を取り出す関数
 7:  *
 8:  *separate(浮動小数点データ,&sign,&exp,&man0,&man1)
 9:  * *sign  ←  符号
10:  * *exp   ←  指数部
11:  * *man0  ←  仮数部（上位）
12:  * *man1  ←  仮数部（下位）
13:  ***********************************************
14:  */
15:
16: void separate(d,sign,exp,man0,man1)
17: double d;
18: int *sign,*exp,*man0,*man1;
19: {
20:   /* &d が double 型へのポインタ */
21:
22:   *sign = (*(int *)&d <0 ) ? 1 : 0;
23:   *exp  = (*(int *)&d >> 20) & 0x7ff;
24:   *man0 = (*(int *)&d & 0xfffff) | 0x100000;
25:   *man1 = *((int *)&d + 1);
26: }
27:
28: main()
29: {
30:  double x;
31:  int sign,exp,man0,man1;
32:
33:  separate(x = 1.2, &sign, &exp, &man0, &man1);
34:  printf("%9.6f S=%d EXP=%03x MAN=%08x%08x¥n",
35:    x, sign, exp, man0, man1       );
36:  separate(x = -2.4, &sign, &exp, &man0, &man1);
37:  printf("%9.6f S=%d EXP=%03x MAN=%08x%08x¥n",
38:    x, sign, exp, man0, man1       );
39:  separate(x = 3.1415, &sign, &exp, &man0, &man1);
40:  printf("%9.6f S=%d EXP=%03x MAN=%08x%08x¥n",
41:    x, sign, exp, man0, man1       );
42: }
```

タ変数の宣言でしたからポインタ変数へのキャストもデータ型の後ろに＊をつけます。

int型へのポインタへキャストするには(int ＊)という演算子によります。よってxがdouble型へのポインタ変数であれば,

＊ (int ＊) x

記述することでxの指し示すdouble型データをint型とみなして参照できます。

このポインタ変数へのキャストを利用すれば，引数の受け渡しという裏技を使ってdouble型データをint型データとみなして処理したリスト3のプログラムをエレガントに書き直せます（リスト7）。

ポインタへのポインタ

ポインタ変数に格納されるアドレス値は符号なしの整数値とみなすこともできます。このため，ポインタ（アドレス値）を要素とする配列を考えることもできます。簡単な例では文字列を要素とする1次元配列，

```
char ＊g []= {
        "gundam","gundam mkII",
        "Z gundam", "gundam ZZ",
        "n gundam"
    };
```

が考えられます。何度も述べましたが，文字列そのものはアドレス値ですから，gという配列の要素は5つのアドレスになります。つまり，

g [0] は"gundam"というアドレス
g [1] は"gundam mkII"というアドレス
g [2] は"Z gundam"というアドレス
g [3] は"gundam ZZ"というアドレス
g [4] は"n gundam"というアドレス

となります。いうまでもなく，宣言のchar ＊という表現（char型へのポインタ）はgの要素がchar型のデータを指し示すアドレス値であることを意味しているのです。このような配列では各要素の指し示す先が実際の文字の並びになります（図7）。

それでは，配列gの要素を指し示すポインタ変数はどう宣言したらよいでしょうか。これはポインタ変数の指し示すものがさらにポインタである場合です。

char ＊p;

はpの指し示すものがchar型データだよという宣言でした。いまはpの指し示したものの指し示すものがchar型データだよというのですから，

char ＊ (＊p) ;

でいいのです。実際の宣言では簡略化し，

char ＊＊p;

と記述します。このように宣言されたpに配列gの先頭アドレス(&g[0]すなわちg)を代入すれば，

```
p     →  &g [0]
p+1   →  &g [1]
p+2   →  &g [2]
p+3   →  &g [3]
p+4   →  &g [4]
```

あるいは，

```
＊ p      →  g [0]
＊ (p+1)  →  g [1]
＊ (p+2)  →  g [2]
＊ (p+3)  →  g [3]
＊ (p+4)  →  g [4]
```

という対応を取ることができます。

もし，ポインタ変数pによってg[2]（すなわち"Z gundam"）の4番目の文字(g[2][3]）を参照したい場合は，

(＊(p+2)) [3]

あるいは，

＊ (＊ (p+2)+3)

という記述を使います。g [0] の最初の文字 (g [0] [0]) を参照するのなら，

＊ (＊p)

すなわち，

＊＊p

です。これはポインタ変数pが2次元配列へのポインタと同等であることを示しています。すなわち，参照を2回行う（＊＊あるいは [ ] [ ] という操作）と目的となるデータ型の要素に行きあたります。

ポインタへのポインタ変数と2次元配列の決定的な違いは，1回目の参照で行きあたるデータの大きさです。2次元配列の場合は1回目の参照で行きあたるものは大きさ（配列宣言時の [ ] [ ] で2つ目の [ ] 内に書かれる値）の定まっている1次元配列です。この1次元配列はメモリ上に連続して並んでいます。ポインタへのポインタ変数の場合はメモリのどこか別の場所に存在する配列で，特に配列の大きさが等しくなければならないという制限はありません。

先の配列gを見てわかるように，それぞれの文字列の大きさ（長さ）は一定ではありません。このように，ポインタの配列やポインタへのポインタ変数は大きさの異なる配列を1次元配列として持つときに便利です。先の配列gを，

```
char g [ ] [12]= {
        "gundam","gundam mkII",
        "Z gundam", "gundam ZZ",
        "n gundam"
    };
```

と2次元配列で記述した場合のメモリの様子を図8に示しておきますので，図7と比較してみてください。この宣言は常識的には受け入れがたいものです。文字列はポインタを表すはずなのに，この宣言ではchar型の配列そのものとして使われています。しかし，通常のコンパイラはこの宣言を，

```
char g [ ] [12]= {
   {'g', 'u', 'n', 'd', 'a', 'm', 0},
   {'g', 'u', 'n', 'd', 'a', 'm', ' ', 'm',
  'k', 'I', 'I', 0},
   {'Z', ' ', 'g', 'u', 'n', 'd', 'a', 'm', 0},
   {'g', 'u', 'n', 'd', 'a', 'm', ' ', 'Z', 'Z', 0},
   {'n', ' ', 'g', 'u', 'n', 'd', 'a', 'm', 0},
  };
```

と正しく解釈してくれるようです（実際私たちもこういう意味で使いたいのです）。

ところで，星が2つの＊＊pというのポインタ変数の宣言があるのですから，＊＊＊pとか＊＊＊＊pというポインタ変数を宣言することもできます。たとえば星3つの，

int ＊＊＊p;

はポインタ変数pの指し示すものが，ポイ

図7　ポインタを使った文字列の格納

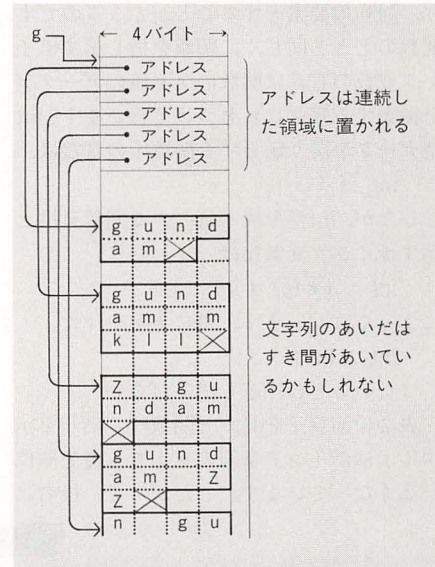
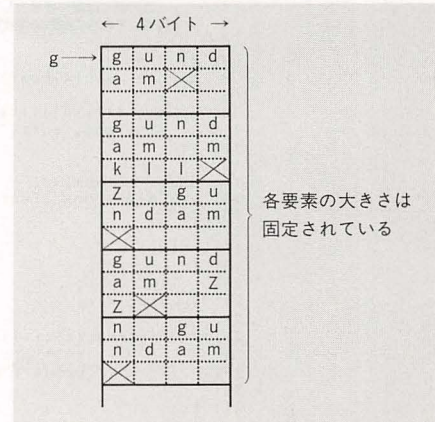



図8　2次元配列による文字列の格納

ンタ（アドレス値）であり，その指し示すものがポインタ（アドレス値）であり，さらにその指し示すものがint型データであるという宣言です。とてもややこしく感じますが，星が3つ以上の宣言を実際のプログラムで見かけることはまずありません（そういえば，なにかのベンチマークプログラムで10個ぐらい＊がつく例を見たなあ）。

関数へのポインタ

C言語では関数へのポインタを定義することができます。関数自身は変数ではないのですが，その実体は変数や配列と同じくメモリ上に置かれていて，その関数の入り口には関数名と1対1に対応するラベルがつけられています。これは，メモリ上のラベルしか存在しない配列名がアドレス値を持っているのと同じことで，関数名もアドレス値を持っていることを示すのです。

このため関数のアドレス値はポインタ変数に代入したり，別の関数への引数としたり，配列の要素とすることができるのです。変数のときと同じく，関数を指し示すポインタ変数の宣言は関数の戻り値のデータ型を指定する宣言に＊をつけて行います。int型データを戻り値とする関数ｆの宣言は，

int f ( ) ;

でしたが，int型を戻り値とする関数を指し示すポインタ変数fpは，

int (＊fp) ( ) ;

によって宣言します。単に＊をつけた，

int ＊fp ( ) ;

ではないことに注意してください。

表2の演算子を見ると関数呼び出しを示す( )はポインタ参照の＊よりも優先順位が高くなっています。したがって，後者はint型データへのポインタを戻り値とする関数の意味になります。前者は＊fpで呼ばれる関数という意味です。

さて，関数へのポインタ変数にアドレス値を与えるには，配列の場合と同じく関数名を代入します。たとえば，ｆという名前の関数があってfpが関数へのポインタ変数である場合，

fp = f;

によって，fpに値を与えることができます。ただ，このとき，またもやコンパイラの都合が顔を出してきます。上の代入の場合，コンパイラがｆを関数と認識していなければなにが起こるかわかりません。ｆは通常の変数かもしれませんし，配列名かもしれません。ｆが関数名であることをコンパイラに教えてやるために，その代入よりも先に関数ｆの定義を記述するか，ｆが関数だよという宣言をする必要があります。

ｆが関数だよという宣言は関数ｆの戻り値のデータ型の宣言で行います。そして，関数を指し示すポインタ変数fpにアドレス値を与えたあと，そのポインタ変数を用いて関数呼び出しを行うためには，

(＊fp) ( )

という記述をします。これによってfpの指し示す関数ｆが呼ばれるのです。

もし，関数呼び出し時に引数を与えたければ，

(＊fp) (a, b, c)

などと( )内に引数を記述すればいいでしょう（a，b，cが引数です）。

もともとC言語では引数のチェックを行わないので，ポインタの指し示す実体（いまは関数ｆ）で定義されている引数を無視した形式で引数を渡してかまいません。リスト8に関数へのポインタを利用したプログラム例を示しておきます。

リスト8

```
 1: /*
 2:      関数へのポインタを用いる
 3:
 4:      この例では配列要素として関数のアドレスを
 5:    持ち、それらの関数を順次呼び出している
 6:
 7: */
 8: double  sin(),cos(),sqrt(),log(),exp();
 9:
10: double (*fp[5])()={ /*関数へのポインタを宣言(ここでは配列)*/
11:     sin, cos, sqrt, log, exp
12: };
13:
14: char    *nam[]={
15:     "sin","cos","sqrt","log","exp"
16: };
17:
18: main()
19: {
20:     int    i;
21:     double r;
22:
23:     for(i=0;i<5;i++){
24:             r=(*fp[i])(3.14159); /* 関数を呼ぶ */
25:             printf("%s(3.14159)=%f¥n",nam[i],r);
26:     }
27: }
```

構造体と共用体

私たちが日常かかわっている対象は多くの場合「構造」を持っています。たとえば，日付は月，日，曜日という構造を持っていますし，個人というデータは名前，性別，身長，体重といった構造を持っています。

この場合の構造とは複数の異なる（同じでもいいが）データが寄り合わさってひとつになっていることを意味します。たとえば，数学では複素数というデータを使用することがありますが，これは実数部と虚数部という構造を持ったひとつのデータです。このように構造を持ったデータのことをC言語では構造体と呼びます。たとえば，

```
struct complex {
    double r;
    double i;
};
```

は複素数を構造体宣言した例です。まずstruct (structure；構造という意味）というキーワードが構造体の宣言を表します。次のcomplexは構造体の名前です。これはintとかdoubleとかいうデータ型の名前に対応しています。その次の{ }でcomplexという構造体の構造を定義しています。この場合，実数部を示すdouble型のｒと虚数部を示すdouble型のｉがあります。このｒとｉが構造体のメンバと呼ばれるものです。

構造体を宣言するとそれをデータ型と同じように使うことができます。つまり，その構造を持った変数を宣言できるのです。このcomplexでいうと，

struct complex x, y, z;

はcomplexという構造を持つ変数ｘ，ｙ，ｚを宣言することを意味します。そして，このときｘ，ｙ，ｚはすべてｒとｉというメンバを持つことになります。

構造体のメンバを参照するためには`.`という演算子を用います。ｘという変数がcomplexという構造を持っている場合，

x.r ：複素数の実数部

x.i ：複素数の虚数部

を表します。いま，このメンバ自身はdouble型として宣言されていますから，これらはほかの式でdouble型データとして使用したり，double型の数値や変数値を代入することができるのです。

たとえば，complexという構造を持った変数ｘ，ｙ，ｚで，ｘとｙを加えてｚに代入するプログラムは複素数の実数部と虚数

部を取り出してそれぞれ足し合わせる,

　　z.r ＝ x.r ＋ y.r ;

　　z.i ＝ x.i ＋ y.i ;

という記述になります。

ところで,上の例ではcomplexという構造の宣言とcomplexという構造を持つ変数の宣言を別々に行っています。しかし,省略がお得意のC言語ではこの2つの宣言を,

```
struct complex {
    double r;
    double i;
} x, y, z;
```

とまとめて記述することもできます。また,プログラムで宣言されるすべての変数(関数の引数宣言を含む)の中でx,y,z以外がcomplexという構造を持たないのであれば,ここでわざわざ,

```
{ double r; double i;}
```

という構造にcomplexという名前をつける意義がなくなってしまいます。そのときはcomplexという名前を省略して,

```
struct {
    double r;
    double i;
} x, y, z;
```

という宣言で十分です。これはx,y,zが,

```
{ double r; double i;}
```

という構造を持った変数ですよと直接宣言するものです。この点,

```
struct {
    double r;
    double i;
}
```

の部分がx,y,zのデータ型を表すものと思ってかまいません。

ですからtypedefによって構造体はデータ型として定義することができます。

```
typedef struct {
    double r;
    double i;
} COMPL ;
```

という記述は,

```
{ double r; double i;}
```

という構造を持ったCOMPLというデータ型を定義しています。先の例のようにある構造に対してcomplexという名前しかついてないのであれば変数を宣言するごとに,

```
struct complex x;
```

などとstructというキーワードを必ずつけなければなりません。しかし,COMPLのようにデータ型として宣言されていると,その構造を持つ変数の宣言は,

```
COMPL x;
```

と簡潔に行うことができます。

さて,これまでの複素数の例は構造体の2つのメンバが同じdouble型をしていました。それならば,2要素からなるdouble型の1次元配列を用いても同じようなことができます。typedefによって,

```
typedef double ACOMPL [2] ;
```

と宣言することでACOMPL型を宣言します。このとき,

```
COMPL x;
```

と宣言した変数xと,

```
ACOMPL y;
```

と宣言した変数yとでは,

　　x.r　→　y[0]

　　x.i　→　y[1]

となる程度であとは大差がないように思えます。しかし,実は決定的な違いがあります。これは構造体と配列の違いです(ACOMPLの実体は配列です)。構造体は別の構造体の内容を直接代入することができます。あるいは,構造体は関数の引数となることも関数の戻り値となることもできます。配列ではそのようなことはできません。

あるいは,構造体は各メンバのデータ型が違っていてもかまいません。しかし,配列は要素(メンバに対応)のデータ型はすべて同じでなければなりません。このような理由で構造体は配列よりも格段に扱いやすいものといえます。また,構造体の場合はメンバを名前で参照できる(配列ならば添字による)こともプログラムの読みやすさを向上させています。

以上のように,複素数は配列でも表すことができるので構造体の例としては面白いものではありません。しかし,乗り掛かった舟です。リスト9に複素数を構造体で実現するプログラム例を示しておきます。構造体が引数で渡されたり,関数の戻り値となっていることを確かめてくださいね。

構造体の構造をアセンブリ言語の観点から考えてみましょう。いま,例として次のような構造体を考えましょう。

```
struct parsonal {
    char   *name; /*  名前   */
    char   age;   /*  年齢   */
    double hight; /*  身長   */
    double weight;/*  体重   */
    double b;     /*  バスト */
    double w;     /*  ウエスト*/
    double h;     /*  ヒップ */
} norip;
```

これは個人に関するデータを示す構造体です。いまはpersonalという構造体の宣言とともに,その構造を持つnoripという変数が宣言されています。変数が宣言されるとその実体がメモリ上に確保されます。この点は,構造を持った変数といってもほかのデータ型の変数と変わりありません。

このとき,メモリ上には各メンバの値を格納するのに十分なだけの領域が割り当てられていきます。たとえば,name(ポインタ変数)のために4バイト,age(char型)

リスト9

```
 1: /*
 2:    関数の引数や戻り値になる構造体
 3: */
 4: typedef struct { double r; double i; } COMPL;
 5:
 6: COMPL cadd(x,y).    /* 複素数の和 */
 7: COMPL x,y;
 8: {
 9:    x.r += y.r;      x.i += y.i;      return(x);
10: }
11:
12: COMPL cmul(x,y) /* 複素数の積 */
13: COMPL x,y;
14: {
15:    COMPL m;
16:    m.r = x.r * y.r - x.i * y.i;
17:    m.i = x.r * y.i + x.i * y.r;
18:    return(m);
19: }
20:
21: COMPL makc(x,y) /* 複素数を作る */
22: double x,y;
23: {
24:    COMPL c;
25:    c.r = x;         c.i = y;         return(c);
26: }
27:
28: main()
29: {
30:  COMPL     a,b,c;
31:
32:  a=makc(1.414, 2.236);     b=makc(3.141, 2.818);
33:  c=cadd(a,b);
34:  printf("(1.414,2.236)+(3.141,2.818)=(%f,%f)¥n",c.r,c.i);
35:
36:  a=makc(1.414, 2.236);     b=makc(3.141, 2.818);
37:  c=cmul(a,b);
38:  printf("(1.414,2.236)*(3.141,2.818)=(%f,%f)¥n",c.r,c.i);
39: }
```

のために1バイト，hight (double型) のために8バイト，……というぐあいです。このときCコンパイラはメンバの表す位置を先頭からのオフセットとして覚えておくのです。この様子を図9に示します。

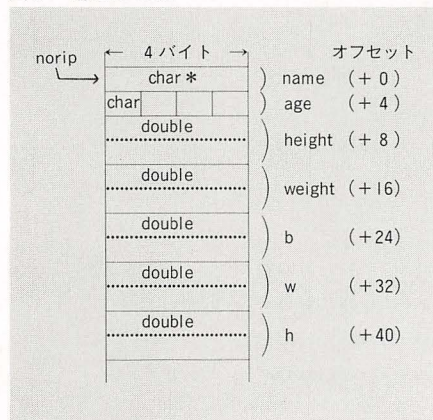
そしてプログラムが構造体のあるメンバを参照するとき，CPUは変数の先頭アドレスにメンバの持っているオフセットを加えたアドレスを参照するのです。つまり，weightというメンバの値を見たいときは，

noripの先頭アドレス　＋　16

番地からdouble型のデータをリードしてくることになります。なお，このときの加算のうち，char型とint型は整数，float型とdouble型は浮動小数点を表します。なお，このときの加算はオフセットを素直に加えるだけです（配列名やポインタ変数ではデータ型の大きさを掛け算して加えていた)。これはCPUでいうとディスプレイスメントつきアドレッシングに相当します。

ところで，図9ではageを表す領域とheightを表す領域のあいだに3バイトのゴミが詰まっていて，heightを表す領域の先頭を切りのいいアドレスになるように調整してあります。実際のCコンパイラではこのゴミはない場合もあります。これはCPUがメモリを参照するとき，もっとも都合がいいように調整されるのです（最小回のバスサイクルでデータを参照できるように置かれる)。実際，68000用のＸＣの場合は1バイトしかゴミが入りません。

このように，構造体の各メンバに対する領域は配列とは異なり，連続したメモリに割り当てられるとは限りません。どこに割り当てられたかを知っているのはコンパイラだけなのです。しかし，プログラムからは各メンバは名前で参照することが原則なので，各メンバに対する領域がメモリ上でどういうぐあいに割り当てられていようがどうでもいいことでしょう。

図9　構造体の例

構造体の初期化

構造体のメモリ上の領域の割り当てを見て，大きさの異なるデータを1次元配列状に並べたものが構造体にすぎないと思った人はなかなかキレる人です。各要素はその大きさが異なるため配列と同様の添字では参照することができません。そこで先頭からのオフセット（メンバに1対1対応する）を添字の代わりに使用しているのです。

なぜこのようなことをいい出したかというと，構造体も配列と同様な方法で初期化できるからです。配列の場合は｛と｝の中に各要素の初期値を順番に記述していくことによって初期化を行います。構造体の場合も同様に｛と｝のあいだに各メンバの初期値を順番に記述していけばいいのです。

前の節で定義したpersonalという構造を持つnoripという変数の各メンバを，

```
name    ←   "Noriko Sakai"
age     ←   18
height  ←   157.0
weight  ←   40.0
b       ←   78.0
w       ←   57.0
h       ←   84.0
```

という値で初期化するためには，

```
struct parsonal norip = {
    "Noriko Sakai",
     18,    157.0,  40.0,
     78.0,  57.0,   84.0
};
```

でよいのです。配列の場合とほとんど変わるところはありませんね。

また，構造体はひとつのデータとして扱うことができますから，構造体の配列も考えられます。構造体の1次元配列（2次元以上の配列も考えられるが見ることはまずない）はイメージ的には配列（構造体のこと）の1次元配列となりますから2次元配列と同等です。したがって，構造体の1次元配列の初期化は2次元配列の初期化と同様に行えます。上の例では，

```
struct parsonal idle [ ]  =  {
  { "Noriko Sakai",
    18,    157.0,  40.0,
    78.0,  57.0,   84.0  },
  { "Mamiko Tanaka",
    15,    156.0,  43.0,
    80.0,  56.0,   82.0  },
  { "Maha Hamada",
    15,    161.0,  46.0,
    78.0,  60.0,   83.0  }
};
```

あるいは，内側の｛｝を省略した，

```
struct parsonal idle [ ]  =  {
  "Noriko Sakai",
  18,    157.0,  40.0,
  78.0,  57.0,   84.0,
  "Mamiko Tanaka",
  15,    156.0,  43.0,
  80.0,  56.0,   82.0,
  "Maha Hamada",
  15,    161.0,  46.0,
  78.0,  60.0,   83.0
};
```

で初期化を行うことができます。

K＆Rでは自動的変数で与えられた構造体についても，配列と同様に初期化できることになっていますが，残念ながらＸＣではサポートしていないようです。

構造体へのポインタ

構造体の配列が考えられるのですから，当然構造体を指し示すポインタ変数も考えられます。構造体を指し示すポインタ変数の宣言は，構造体の宣言で変数名の前に＊をつけることで行います。これは通常の変数や配列を指し示すポインタ変数の宣言と同様です。たとえば，

```
struct personal *idle;
```

とはpersonalという名前を持つ構造体を指し示すポインタ変数idleを宣言することを意味します。また，普通の変数と同じく，構造体であると宣言されている変数は＆演算子によってその先頭アドレスを取り出すことができます。

逆にポインタ変数からそれが指し示す構造体のメンバを参照するには，やはり，演算子を使用します。上のように構造体を指し示すポインタ変数idleが宣言されているとき，構造体personal（前節で定義したのと同じとして）の各メンバの参照は，

```
(*idle).name
(*idle).height
```

などという記述で行います。

しかし，構造体を指し示すポインタ変数からメンバを参照することはしばしば行われるためか，そのために特別な演算子が用意されています。それが，－＞という矢印みたいな演算子です。この演算子を使えば上のメンバの参照は，

```
idle -> name
idle -> height
```

と記述することもできます。この－＞という記号はほかのプログラミング言語で見か

けることがありませんから，この記号を多用しているといかにもC言語を使っているという気になりますね。

関数への値の受け渡しのほかに，構造体を指し示すポインタ変数がよく用いられるのは線形リスト（大きさが決まっていない配列みたいなもの）の構造を表現するときです。これは図10に示すようにある要素(構造体）がポインタによって次から次へ連ねられたものです。各要素は必ず次の要素を指し示すポインタを持っています。そして，最後の要素のポインタは「どこも指し示していない」という印になっています。

ところで，配列の宣言では配列の大きさ（[ ] の中で宣言するやつ）を同時に宣言しなければなりません。このため，配列の大きさを越えて配列要素を持つことはできません。すなわち，宣言した配列の要素を使い切っているとき，さらに新しい要素が必要になってもどうしようもありません。しかし，線形リストには大きさという概念がありませんから，簡単に新たな要素をつけ加えることができます。このときは，新しい要素を用意して，現在の線形リストの終わりの要素のポインタをその要素を指し示すように変更するだけでいいのです（図11)。この点，線形リストは配列よりも柔軟性があります。線形リストを宣言する具体的な例は次のようなものです。

```
typedef struct _lis {
  struct _lis *next; /*次へのポ
                        インタ  */
  int       num;  /*会員番号*/
  char      *name; /* 名前 */
} LIS ;
```

これは線形リストのひとつの要素のデータ型LISを定義しています。この定義で，

struct _lis *next;

の部分が次の要素へのポインタの宣言です。この定義では_lisという構造体を宣言するのに_lisを参照しているように見えます。しかし，構造体の中で宣言しているのは_lisへのポインタ(4バイトの領域で_lisの構造には無関係）であって，_lisの構造ではありません。Cコンパイラは_lisという構造体が存在するんだよということさえわかっていればいいのです。上の例ではnextというポインタを宣言する1行上の，

typedef struct _lis {

という部分で_lisという構造体があることがわかっているのでCコンパイラに混乱はなにも起こりません。リスト10にここで定義した線形リストを操作するプログラムを示しておきます。

共用体とは

C言語では構造体によく似たデータ型として共用体が使えます。共用体の宣言の例は次のものが考えられます。

```
union data {
    char  c;
    short s;
    int   i;
  }   D;
```

これはdataという共用体の宣言で，その構造は{ }内で示されています。構造体の

図10　線形リストの例

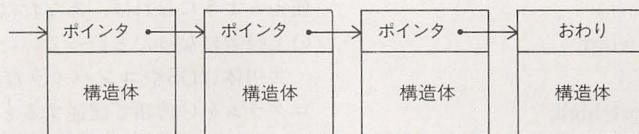
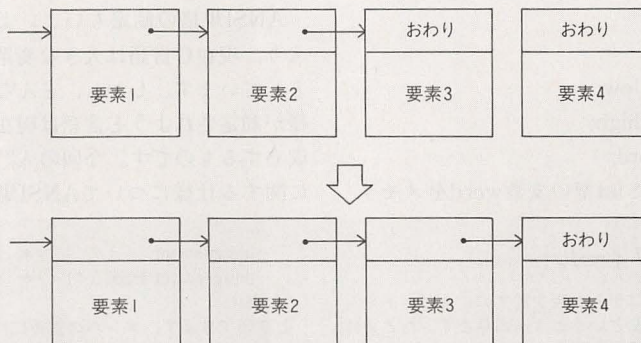



図11　要素の追加

リスト10

```
 1: /*
 2:      線形リストを扱うプログラム
 3: */
 4: #include <stdlib.h>            /* malloc を使う */
 5: typedef struct _lis {          /* LIS 型の宣言 */
 6:         struct _lis *next;
 7:         int     num;
 8:         char    *name;
 9: } LIS;
10: /*
11:    用意した関数
12:
13:    LIS  *make_cell(int num, char *name)要素を作る
14:    void append (LIS *L, LIS *E) Lの最後にEを追加する
15:    void print_lis(LIS *L)         Lをプリントする
16: */
17: LIS *make_cell(nu,na)
18: int  nu;
19: char *na;
20: {
21:     LIS *ep;
22:     ep = malloc(sizeof(LIS));      /* 領域を取る */
23:     ep->next = 0;   ep->num  = nu;  ep->name = na;
24:     return(ep);
25: }
26:
27: void append(lis,e)
28: LIS *lis;
29: LIS *e;
30: {
31:     while(lis->next) lis++;       /* 終わりを探す      */
32:     lis->next = e;                            /* ポインタのつけかえ */
33: };
34:
35: void print_lis(lis)
36: LIS *lis;
37: {
38:  int c=0;
39:  while(lis->next){
40:    printf("[ (%d)  %s ]→", lis->num, lis->name);
41:    if(++c == 3){ c=0; printf("¥n");}
42:    lis = lis->next;                     /* 次の要素 */
43:  }
44:  printf("[ (%d)  %s ]¥n¥n",lis->num,lis->name);
45: }
46:
47: LIS *root;
48:
49: main()
50: {
51:   root = make_cell(1,"アムロ レイ");
52:   print_lis(root);
53:
54:   append(root, make_cell(2,"カミーユ ビダン"));
55:   print_lis(root);
56:
57:   append(root, make_cell(3,"ジュドー アーシタ"));
58:   print_lis(root);
59:
60:   append(root, make_cell(4,"クリス マッケンジー"));
61:   print_lis(root);
62: }
```

宣言と比べるとstructというキーワードがunion(連合の意)に置き換わっただけです。メンバ参照も構造体とまったく同じです。

ただ構造体と異なるのは構造の違いです。先に構造体のメンバはメモリ上における先頭からのオフセットを示していると説明しました。共用体とはこれの特殊な場合でオフセットがすべて0として扱われるものをいいます。つまり，上のdataという共用体のメンバはメモリ上の同じ位置を示しています(図12)。すなわち，c(char型)というメンバなら1バイト分，s(short int型)というメンバなら2バイト分，i(int型)というメンバなら4バイト分のデータをその位置から読み書きできるのです。

共用体は構造体と組み合わせて用いられることが多いようです。たとえば，

```
union {
    int      word;
    struct {
        short high;
        short low;
    }        hword;
} REG;
```

という宣言は，

```
struct {
    short low;
    short high;
} hword;
```

という構造体とint型の変数wordをメモリ上で共有することを指定するものです。このような構造を持つ共用体として宣言されたREGという変数は(それが割り当てられたメモリ領域に対して)，

REG.word

で全体を参照することもできますが，

REG.hword.high

によって上位16ビットを，

REG.hword.low

によって下位16ビットを参照することができるようになります。

しかし，これは構造体や共用体のメンバがメモリ上でどのように領域を割り当てられているのかを知っていなければなかなか活用できるものではありません。これこそコンパイラのクセやアセンブラの知識をもろに使用するものであり，共用体が自由に使えるようになれば，あなたはもうC言語の上級者になったといっていいでしょう。

共用体はOSやコンパイラなどの基本プログラムをC言語で記述するときに使用することがありますが，構造体とは異なり，あまり使用されることはありません。

＊　＊　＊

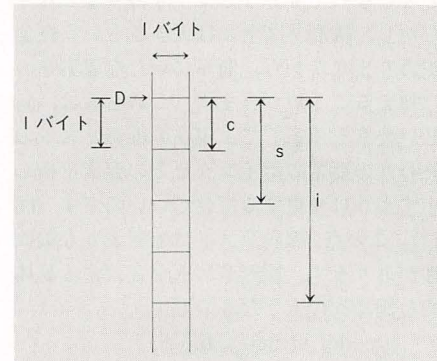
ANSI規格の制定もいよいよ最終段階に入り，現在C言語は大きな変革を迎えようとしています。しかし，どんなに立派な仕様が制定されようと言語は現在の延長上に成立するものです。今回の入門ではC言語に関する仕様についてANSI規格をある程度意識しましたが，それと従来の仕様を特に区別はしていません。原則的にはXCを標準としてありますが，便利だと考えられるものはANSI仕様を適宜取り込んで説明してきました(GCCでは-fraditional オプションが必要な場合があります)。

少し長めの原稿になりましたが，高級アセンブラと呼ばれるC言語に対して私が感じているイメージをなんとか表現できたと思います。やはりなにかを学ぶ場合，形(文法)だけを覚えても応用がききません。その背後にある概念や意義を自分なりに把握することが大切なのです。そういった考え方を身につける手助けになれば幸いです。

参考文献

1)　カーニハン，リッチー，「プログラミング言語C 第2版」，1989年，共立出版

2)　米田(編)，「C――言語とプログラミング――」，産業図書，1982年．

図12　共用体の例

ビットフィールド

世の中にはデータ長を表すのに4ビットや1ビットで十分というときもあります。たとえば，10進数の1桁を表現するためには4ビットで十分ですし，なにかの状態のオン/オフなら1ビットで十分です。4ビットや1ビットで表現可能な変数のためにわざわざ8ビットもメモリを使うのはもったいないと思う人がいるかもしれません。このような人のためを思ってかどうか知りませんが，C言語ではビットフィールドというデータを使えるようになっています。

これは任意のビット長(といっても，その上限はCコンパイラによってまちまち)を持った整数データです。このビットフィールドは構造体のメンバとして宣言できます。ビットフィールドの宣言は通常の構造体の宣言と同様な形式で行いますが，各メンバの名前の後ろにコロン(:)をつけてビット長を指定します。

たとえば，68000のCPUのステータスレジスタはビットフィールドを用いて，

```
struct {
unsigned int c :1;/*  キャリー  */
unsigned int v :1;/*  オーバーフロー */
unsigned int z :1;/*  ゼロ  */
unsigned int n :1;/*  ネガティブ  */
unsigned int x :1;/*  拡張  */
unsigned int   :3;/*  ダミー  */
unsigned int imask :3;/* 割込みマスク */
unsigned int   :2;/* ダミー */
unsigned int super :1;/* スーパーバイザ */
unsigned int   :1;/*  ダミー  */
unsigned int trace :1;/*  トレース  */
} SR;
```

と表現できます。メンバの宣言についたunsignedはこのメンバ(ビットフィールド)をほかのchar型やint型の整数に代入するとき符号拡張を行うなという指定です。unsignedがついていなければ代入のとき符号拡張されてしまいます。

メンバの名前がない宣言はただそのビット長の領域だけ確保するための宣言です。この領域はあとから参照することができません(メンバ名がないので当たり前)。ビットフィールドは原則的にはメモリ上に1ビットずつ順番に領域を割り当てていきますから，上のSRという変数は16ビットの領域を占めます。それでいて，

```
SR.x=0;
SR.imask=3;
ov=SR.v;
```

などと構造体の普通のメンバと区別なく使用することができますから便利だといえるかもしれません。しかしCPUにとってみれば，バイトアドレスしかないメモリをビット単位に参照しようというのですから実行時間がかかってしまうことを考慮に入れねばなりません。

経験的には，ビットフィールドは先に示した共用体とペアで用いることが多いようです。たとえば，上の例のフラグ部分をBという構造体で宣言し，それをchar型のCCとunionでつないだ構造をSRという変数で宣言します。

```
union {
char  CC;      /* フラグ全体 */
struct {
  unsigned int c :1;/*キャリー  */
  unsigned int v :1;/*オーバーフロー */
  unsigned int z :1;/*ゼロ  */
  unsigned int n :1;/*ネガティブ  */
  unsigned int x :1;/*拡張  */
  } B;
  } SR ;
```

フラグ全体を参照したいときにはCCを使用して，

SR.CC

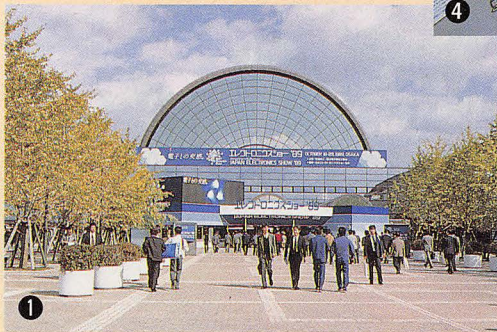
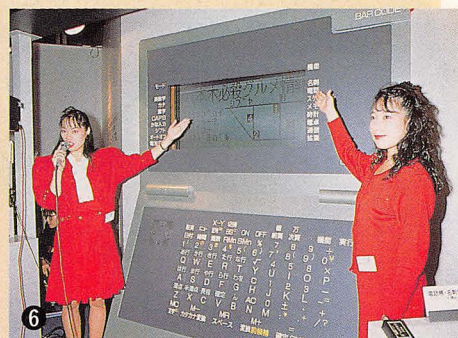
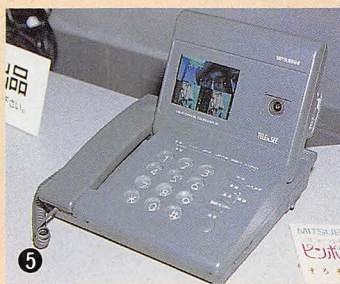
で行えますし，個々のビットを参照したいときはBのほうを用いて，

SR.B.z

などとできるからです。

ビットフィールドの実現はANSI規格でさえも各コンパイラに任せられている部分が多い不明確な部分です。ビットフィールドはメモリ上ではワードデータを基本としてその端から順に割り当てられていきますが，それが左(MSB側)からか，右(LSB側)からかはコンパイラ任せの最たるものです。これは特に共用体とともに用いられる場合(この場合が圧倒的に多い)，C言語の異機種間の移植を困難にする第一の原因となっています。

ですから，ビットフィールドや共用体を用いる場合はできるだけわかりやすく(あるいは，移植しやすいように)プログラムを書いて移植時の妨げにならないようにしなければなりません(もっとも，個人的なプログラムではなにをやっても許されるのですが)。

エレクトロニクスショウ
ELECTRONICS SHOW '89

❶エレクトロニクスショウの会場(インテックス大阪)
❷クリアビジョンは今年の注目商品
❸今年のシャープは液晶一筋
❹シャープのISDN対応TV電話（参考出品）
❺三菱のカラーTV電話（参考出品）
❻NECのバーコード対応電子手帳（参考出品）
❼シャープのもうひとつの目玉，電子手帳
❽コンパニオンのハイライトはビクター
❾エプソンのハンディ翻訳機
❿NECのFM多重受信機（参考出品）

AVのエレクトロニクスショウ'89

10月19日からインテックス大阪で開催されたエレクトロニクスショウ'89は，去年同様クリアビジョン，液晶などが目を引いた。

ホームシアターが作れそうなシャープの100インチ液晶ビジョン。日立の41インチ投影型プロジェクションTV。8月24日に本放送が開始されたクリアビジョン（EDTV）対応のTV。本格的な大画面・高画質時代が到来していることを実感させられた。

クリアビジョンは，NTSCと互換性を持ちつつ画質を向上する方式だが，普通のテレビ2台分とまだ価格は高い。ハイビジョン（HDTV）に至っては数100万円也，個人レベルとはほど遠いという印象だ。

「きれいやな，うちも液晶にしたろ」とは地元関係者らしき人。シャープの14型・6型TFT液晶TVに見とれているのだ。確かに，CRTに比べてちらつきもなく鮮やかな画面である。これならCRTにとって代わることは間違いない。また，「液晶のシャープ」以外の各メーカーも液晶ディスプレイを使った製品をこぞって展示しており，液晶の爆発的な普及が予想される。今回のシャープのメインである，カラー液晶ファインダー内蔵カメラのデモも非常に盛況であった。

カラー液晶TV電話も目を引いた。シャープはISDN対応の製品を参考出品（256×240画素の画像を毎秒4枚送る)。一方，三菱電機は電話回線で使用できる製品を参考出品していた（100×160画素の画像を7.5秒で送る)。これは，個人的に欲しい製品であった。

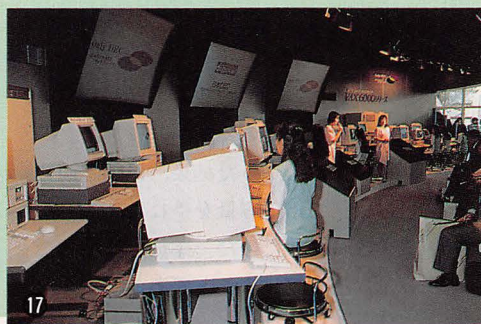
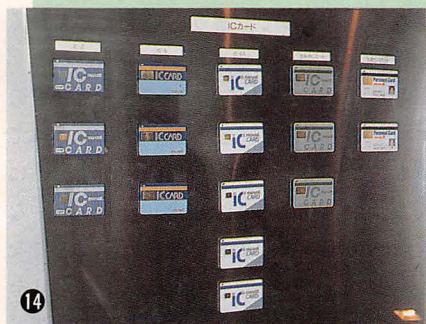
最近のヒット商品，電子手帳も各社の主力製品だ。200万台出荷のシャープに続けとばかりに，NECではバーコード名刺が読み取れるバーコードリーダー内蔵の電子手帳を参考出品。横ではバーコード名刺研究会という団体も普及活動を行っていた。

他には，ビクターが，超小型のS-VHSCビデオコンポを参考出品。NECは，パソコンで制御できるS-VHS VTRの新製品を参考出品。エプソンは，自らサイバートランスレーターと呼んでいるハンディ翻訳機を出品。また，各社から，FM電波の文字・音声・図形などの情報を受けるFM多重受信機が参考出品されていた。

情報機器のデータショウ'89

データショウ'89は，10月24日から，今年も東京晴海の国際見本市会場で開催された。今年は，32ビットパソコン，RISCマシン，ICカード，ISDN対応製品など多数が出品され質量ともに充実していたといえる。

データショウ

DATA SHOW '89

⓫データショウの会場風景（東京：晴海）
⓬NECのブックサイズ98
⓭X68000とCYBERNOTE（参考出品）
⓮日立マクセルのICカード
⓯NTTデータのマルチメディアパソコン
⓰以外と地味なIBM PS/2 486のデモ
⓱データショウっぽいDECのブース
⓲ポータブルMacintosh
⓳飛行機内風の東芝DynaBookのデモ
⓴オムロンのハイレグコンパニオン

NECのブースでは，話題のブック98やNESAバスのPC-H98など新製品が目白押し。また，アクティブマトリクス方式TFTカラー液晶のラップトップPC-9801を参考出品。これは，LX5CなどのSTN方式に比べ美しく高速。——32ビットカラーラップトップPC-9801も間近かと感じた。

われらがX68000は，どこにいるかというと……いました，いました。今年の展示は，常駐文具ツールのStationery PRO-68Kと電子手帳用データベースCYBERNOTE PRO-68K。CYBERNOTEは，参考出品らしいが，チラシにはしっかり19,800円と記入してあったので近々販売されるのではないか。電子手帳所持者必須のアイテムだ。

目新しい記憶装置として，マクセルのICカードやRAMカード，外部記憶装置としても使えるビクターのDAT，映像も共用できるパイオニアの光ディスクなど各種が出品されていた。今後の展開が注目される。また，今年，運営が開始されたISDN対応の製品も多数出品。

IBMは，486登載のPS/2を出品。地味なデモながら人が多く，資料を貰うのに一苦労。ソニー・テクトロニクスやオムロンはMC88000マシンを出品。特にオムロンは4CPU(60MIPS!）で分散OS,Machをサポートする。DECは，ミニコンとEWSの両方をアピール。アップルは，ポータブル型Macintoshを出品していたほか，MacでDTV環境を実現するボードも参考出品。Macのグラフィック環境のすごさを再認識した。

しかし，アップルにしてもDECにしても，外資系の企業はプレゼンに力を入れていることを実感。日本企業では，飛行機のセットを組んでDynaBookの操作デモを行っている東芝とオムロンが目を引く程度であった。

なにか人だかりがしているので覗いてみると，おっと水着のコンパニオン。今回のデータショウはモーターショウと重なったこともあって，良質な（失礼！）モデルは取られてしまった，と同行したカメラマン氏もボヤいていた。やっと絵になるモデルを見つけた我らは，これで，読者からのクレームもこないだろうと安堵した。

ネットワーク志向

今年は両方のショウから，ネットワーク志向が感じられた。今年はEWSだけではなく，パソコン，電子手帳，TV電話など各種のスタンドアロン機器とのデータ交換をより積極的に行おうという動きが著しい。情報関連機器が，一般ユーザーにとっても身近になってきたということだろうか。 (S)

●試用レポート

X68000専用ハードディスク

IT X640/680

左がIT X640, 右がIT X680

10月号のペンギン情報コーナーでお伝えしたように，アイテックからX68000用ハードディスクの新製品が発売された。記憶容量40MバイトのIT X640と80MバイトのIT X680である。今回はこれらの製品について試用レポートをお届けしよう。

●

X68000に対応したハードディスクが目立ってきている。値段もひと頃に比べれば驚くほど下がり，X68000の世界でもハードディスクはだんだんと標準的な外部記憶として市場を形成しつつあるようだ。今回，IT X640とIT X680(以下X640とX680とする）を発表したアイテックは，以前からX68000対応のハードディスクユニット（IT X203とIT X403,)を供給してきた。

X68000とPC-9801のハードディスクのインタフェイスは基本的に互換性があるので，PC-9801用に売られているハードディスクのなかには，ドライブを流用することができる機種も多い。それでも，ものによってはたまにX68000では動作しなかったり，動作が保証されなかったりするので，この機種のようにメーカーがはっきりと「X68000対応」を保証しているのは嬉しいことである。

デザインが第1のチェックポイント

さっそく，製品紹介に入ろう。

まず,「X68000対応」らしさがいちばんよく出ているのが，デザイン。X68000専用といってもよい色は，例によってブラックとグレー。デザインは機能には直接関係しない部分であるが，デザインも売りのひとつであるX68000である。その横に置くハードディスクのデザインは重視されても不思議ではない。その意味では，スタイルそのものはX68000と統一されたデザインとは必ずしもいえないが，それでも従来機種(X203とX403)から一新され，かなりいいデザインになったと思う。サイズも少し小さめになったし，軽量にもなった。

大容量のドライブ環境のために

今回のモデルチェンジでいちばん変わったところといえば，記憶容量である。従来機種の容量は，20Mバイトと40Mバイト。新機種のそれは，40Mバイトと80Mバイト，それぞれ倍になってのラインアップだ。ドライブの値段が下がった結果，以前と同じくらいの価格で倍の容量を手にすることができるようになったのだ。今後も標準的な記憶容量は増え続けることであろう。ま，個人で使うぶんには40Mバイトでも十分広いが，将来にわたって使いたい向きには，80MバイトのX680も決して高い買い物ではないだろう。ただし，大容量のハードディスクを100%活用するためには，Human68kのVer.2.0の使用を勧めたい。

Human68kのVer.2.0では，1ドライブあたり40Mバイトまでをサポートすることになっており，X680の場合，80Mバイトをまるまるひとつのドライブとしては使えず，40Mバイト×2と，見掛け上は2つのハードディスクを並べた格好になっている（しかし，将来的にHumanのバージョンアップがあって40Mバイトを超える外部記憶のサポートがされたときは，内部の配線を変更することで80Mバイトドライブにもなるとマニュアルには断ってある)。

だから，SWITCH.Xで，接続するハードディスクの台数(HD_MAX)を設定するときは，筐体が1台でも，「2」と設定しないと片方の40Mバイトものドライブが死蔵されることになってしまう。いうまでもなくX640のほうは「1」でよい。

そして，Ver.2.0使用時のみ，8台までのユニットを増設して使うように設計されている。これにより，さらに広い領域が手に入る。すべてX680（80MBユニット）なら,計640Mバイトまで可能ということになる。また，従来機種であるX403(40MB)をまぜて使うこともできる。もちろん内蔵ドライブを持つACE, EXPERT, PROのHDタイプにも問題なくつながる。

ターミネータが必要

ひとつ注意しておかなくてはならないのは，ターミネータ（終端抵抗）のことだ。これは，複数のハードディスクユニットを増設する設計になっている場合には欠かせないものである。ユニットを何台つないでいるのかを，メモリスイッチではなく，ハードウェア的にコンピュータ本体に知らせるためのものと思ってもらえばよいだろう。

X640/X680の背面にはコネクタが2つずつ付いており，それらの間をケーブルで次々とつないでいくことになる。そして

最後のユニットのコネクタにはターミネータをつけて，もうそこから先にはユニットがないことを示すようになっている。

ターミネータは，1台だけで使う人も，「この1台でユニットは終わり」という印なのだから，当然つけなくてはならない。このハードディスクのシリーズでは，ターミネータは外付けであり，内蔵されていない（別売りで4,000円)。大方のユーザーはこのユニット1台きりで使うに違いないので，ターミネータは内蔵にして，2台以上つなぐ場合はスイッチ切り替えなどで対応してほしかった。

ターミネータをうっかりつけ忘れて起動してもなぜか動いてしまったりするのが危ないところなのだが，こんな綱渡り的な使い方では保証が受けられなくなる可能性もあるので，メーカー推奨の使い方は厳守すべきだろう。

フォーマットする

接続が終わったらフォーマットである。フォーマットのやり方その他は，本誌9月号の特集を見ていただいてもいいし，マニュアルを見てもいい。このマニュアルは，メモリスイッチの設定から領域確保まで手とり足とり説明してあるので，初めての人でも簡単に使い始められる。もちろん従来機種のマニュアルでも解説してあったが，今度はHuman68kのVer. 2.0用の解説がある。こういうところもまたX68000対応機種の強みではある。

X68000対応といえば忘れてならないのがOS-9。マニュアルには，OS-9/X68000でのフォーマット方法も新しく追加された。

特記すべきは，X680のフォーマットの速さである。シーク時間・データ転送速度とも従来機種やX640よりも速い（平均シーク時間が28→20msになり，データ転送速度が5→10Mbit/秒と倍になった）X680は，フォーマットと領域確保のときに威力を発揮する。40Mバイト×2の広大な領域のフォーマットでも，思ったほど待つ必要がなかった。

マニュアルに載っているのは基本的にフォーマットまで。環境の構築のしかたや，ハードディスクをおいしく使う知恵などについては，9月号の特集がきっと力になってくれることだろう。

使用感覚は……

それでは，リセットボタンを押して，実際に使ってみることにしよう。

使い勝手は，実のところそれほど変わらない。ほかより速いはずのX680にしても，劇的に速くなったという感じはしない。機械の性能はカタログスペックの数字だけでは決まらないことをつくづく感じた次第である。Human68kの立ち上がる時間やコンパイル時間，グラフィックデータのロード時間まで調べてみたが，ほとんど差はつかなかった。どうやら処理時間の大半を占めているのはディスクアクセス以外の部分のようである。

フロッピーディスクのディスクアクセス時間は処理のかなりの時間を喰うので，フロッピーからハードディスク（もしくはRAMディスク）に移ったときはえらく速くなる。ところが，いったんハードディスクを使い始めると，高価な機種を使うことは，処理が何倍にも速くなるということを必ずしも意味しない。まあ，10Mbit/秒といえば，単純計算でもフロッピー1枚の内容を1秒で転送してしまうほどのスピードであることを考えれば，驚異的に速くなると考えるほうがおかしいともいえるが。

振動に対する強度

最後に，ハードディスクの宿敵は振動であるが，その対策はどうだろうか。従来機種（X203/403）には電源を落とすときに自動的にシッピングする「パーキングファンクション機能」なるものがあり，電源を切ると，カシャンという音が意外に大きく響く。ヘッドがディスクから離されて機械的に固定されるためで，こうなれば少々（かなり？）の衝撃にも平気になる。

しかし，この音が不評だったのだろうか，X640/680からはスイッチを切ってもその手の音がしなくなるとともに，マニュアルからもパーキングファンクションの文字が消えた。が，心配ご無用。BREAKキーを押したときにセレクトランプ（正面についている2つのランプのうち赤いほう。いわゆるアクセスランプ）が一瞬点灯する。ごく標準的なシッピングを取り入れたようだ。

*

まとめると，IT X640は標準的なスペックのハードディスクで基本性能は充実している。IT X680は高速かつ大容量のハードディスクだ。飛び抜けたコストパフォーマンスを持っているわけではないが，X68000の専用機として発売されたところに，今回の新製品の意義があるだろう。

ハードディスクの使用によって得られる環境は人間の感覚を簡単に変えてしまうくらい魅力的だ。この私など，ハードディスク使用歴がたった2カ月あまりのくせに，ハードディスクにちょっと染まったくらいでもう，フロッピーのみで使っていた数カ月前のことなどころっと忘れて，「ハードディスクは標準的」などと言っているのである。いい気なものだ。

IT Xシリーズの基本仕様

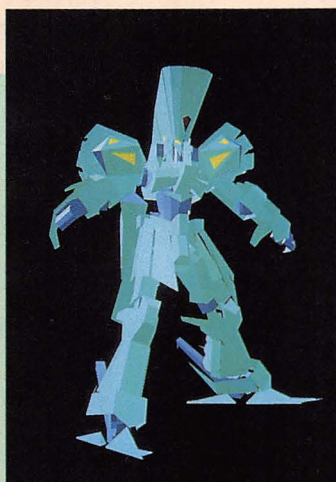
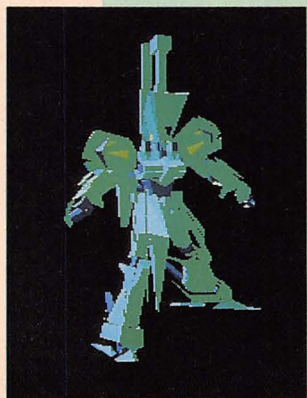
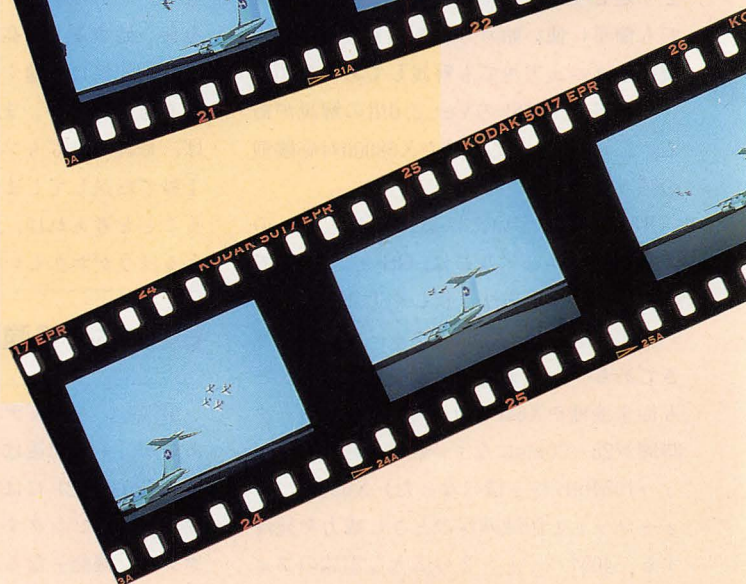
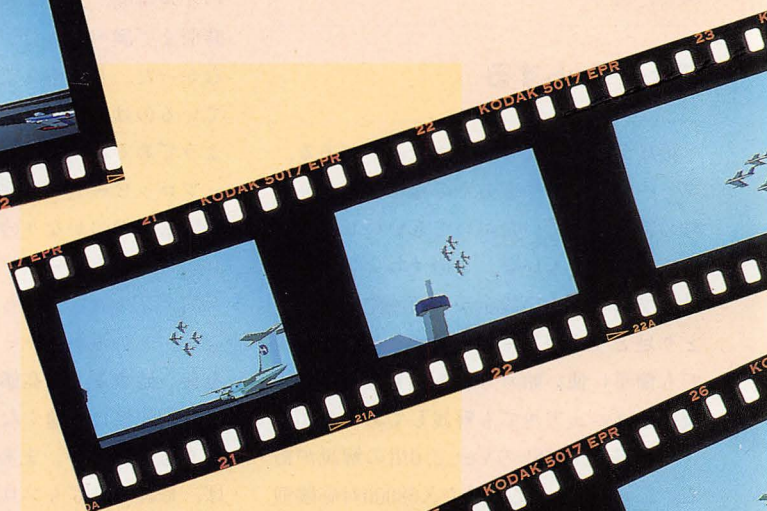
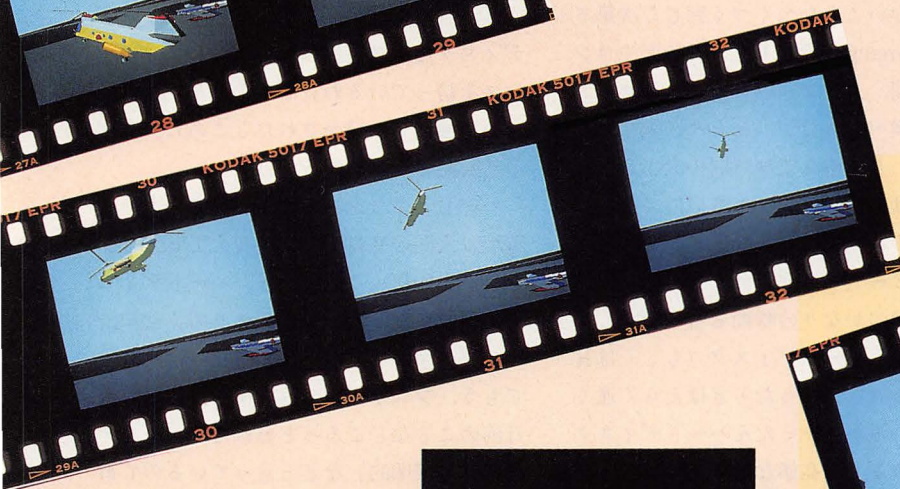
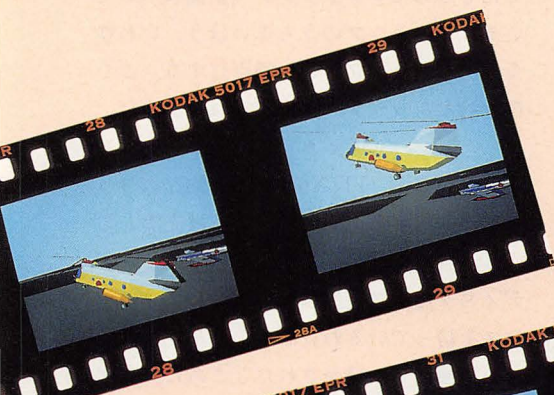
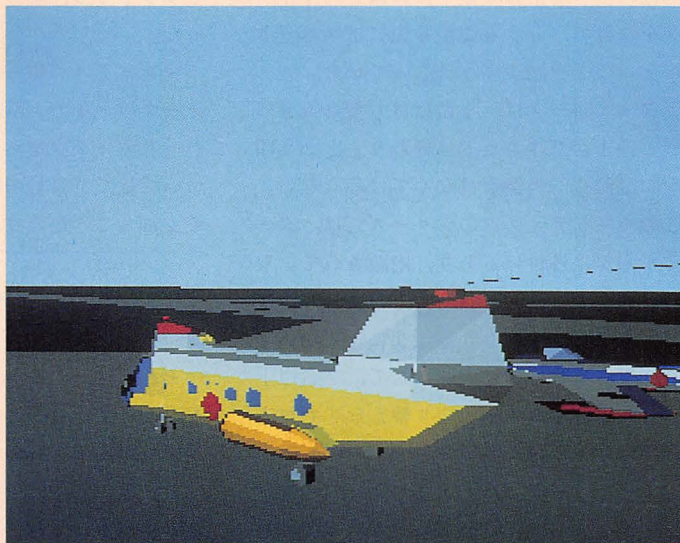
製品名		IT X640	IT X680
記憶容量（フォーマット時）			
装置	(MB)	42.0	85.8
シリンダ	(KB)	65.536	92.16
トラック	(KB)	8.192	18.432
セクタ	(バイト)	256	512
データ転送速度	(Mbit/秒)	5	10
アクセスタイム			
トラック間	(ms)	8	7
平均シーク時間	(ms)	28	20
最大シーク時間	(ms)	54	39
ディスク回転数	(rpm)	3,600	3,600
外形寸法	(W×H×D)mm	87×142×260	87×142×260
重量	(kg)	2.4	2.3
消費電力	(W)	28	35
価格	(円)	158,000	198,000

Oh!X Graphic Gallery

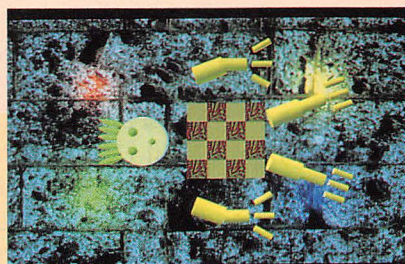
DōGA・CGアニメーション講座

「寺田の教育的指導」

今回ご紹介するのは，村上公三さんの作品「美保基地（米子空港）上空を飛ぶF86Fブルーインパルス」と「飛び立つKV107」です。特に前者はF86Fの動きとカメラワークのからみがダイナミックで，抜粋された写真ではそのリアリティがお伝えできなくて残念です。どうか思いっきり想像力を働かせてご覧ください。

さて，もうひとつ村上さんが送ってくれたのがCADのモデリング機能をフルに使ったロボットもの。かっこいいですね。なお，寺田氏の解説は110ページです。

アンス・コンサルタンツ

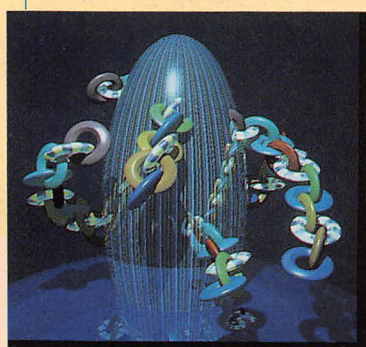
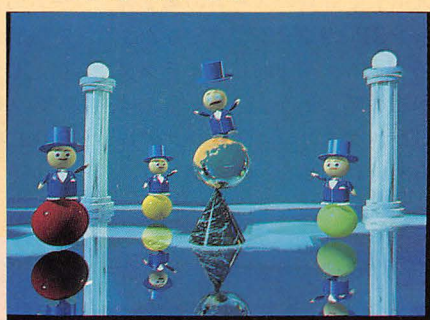
第1回サイクロンCG大会

レイトレーシングソフト「サイクロン」でお馴染みのアンス・コンサルタンツ主催のCGコンテストが開かれました。第１回とはいえ見てのとおりの力作揃い。モデラー高津氏の解説（117ページ）と併せてご覧ください。

モデリング賞
「BIKE」 橋元　弘司

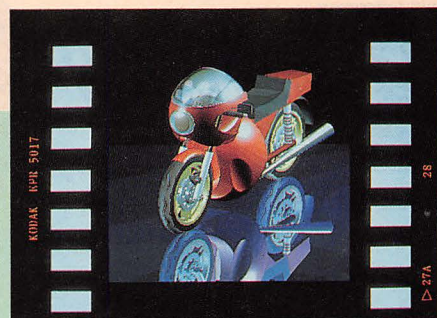
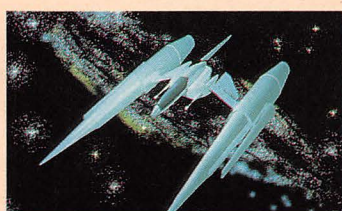
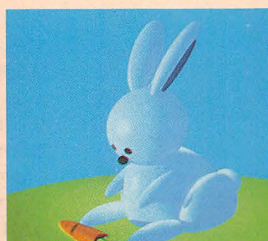
グランプリ賞「SALOON」 益津　享

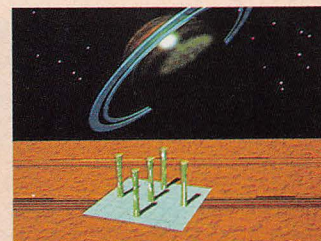
レンダリング賞
「ダンス」 田淵　友章

LOGIN賞
「レイトレックに負けるなよ」 江畑　一

LOGIN賞
「CHILD」 塚田　哲也

Oh!X賞
「propose under the moon beam」 菊池　彰

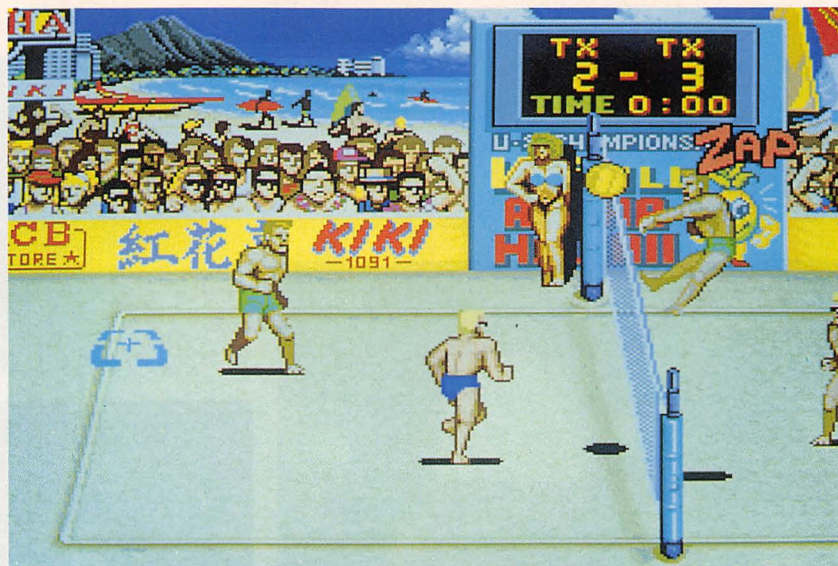
SHARP賞
「ワキウリ」 富田　保男

SHARP賞
「TAMANORI」 駒切　正

「AVERAGE PEOPLE」
阿山　修

「スタースピーダー」 山崎　誠

「欲望の適応」 原　志津雄

「幻想の銀河中心」 渡久山　朝賢

「惑星」 柳沢　学

THE SOFTOUCH

SOFTWARE INFORMATION

SOFTWARE information

V'BALL　滑り込みのレシーブや，高い位置でのジャンプアタックなど，選手の動きが結構リアルで面白い。それに強い球に当たってズズーンと倒れ込むさまも，砂に足を取られるビーチバレーならではというカンジが出ていてよいのです。

話題のソフトウェア

年末年始に向けて，今回は非常に多くのソフト情報が入ってきました。さっそく順を追って紹介していきましょう。

まずは，コナミのシューティングゲーム**A-JAX**。グラフィックもゲーセン版とそっくりで，なかなか期待の持てる一作です。

続いては，電波新聞社の**モトス**。要は敵をはじき飛ばすという単純明快なルールのゲームです。4年ほど前ゲーセンでやった人には懐かしいゲームでしょうね。

「ジェノサイド」で評判のズームの第2弾は，アクションRPG**ラグーン**です。なんといってもこのゲーム，全体的にキャラクタが大きいのが特徴でしょう。RPGと言えば，新規参入会社ヘルツエンジニアリングの**レナム**も忘れちゃいけません。グラフィックもなかなかだし操作が簡単なわりには，のめり込めそうなゲームですよ。そうそう，システムハウスオーからもアダルトRPG**アリスたちの午後**が出るようです。

さて，次はアクションにまいりましょうか。デービーソフトの**フラッピー2**。あいかわらず愛敬を振りまいてくれます。後半面になると結構難しくてヤリがいがありますよ。ボスキャラとの戦いも，楽しく仕上がっています。ワイヤーフレームの新鮮さを改めて感じさせるコムパックの**GAMMA PLANET**も面白そうです。

野球好きの人なら見逃せないのがビクター音楽産業の**やじうまペナントレース**。チームの監督となって采配を振るい，チームを優勝に導くのが役目ですが，もちろん選手としてもゲームに参加できます。そのほかのスポーツものとしては，シャープの**V'BALL**なんかもやっているうちに，つい力が入りそうだし，発売されたばかりのアート

トップ争いは混戦模様

		(前回順位)
1	アフターバーナー	3
2	ジェノサイド	1
3	ファンタジーゾーン	2
4	ローグ・アライアンス	9
5	ソーサリアン	7
6	テトリス	4
7	スタークルーザー	8
8	リングマスター1	6
9	サバッシュ	5
10	Wings	—

今月は首の差でアフターバーナーがトップ奪回！　もうTOP10に登場して半年ですが，トップに返り咲く体力が残っているとは大したもんです。「友達うけが一番いい」，「サイバースティックがあるから」など，さまざまな声が聞かれましたが，なかには「X68000にしかないソフトだから」というのもあって，笑ってしまいました，はっはっ。

しかし，迫力に定評のあるジェノサイド，出来のよさ＋オマケ（みんな，ハリアー面はもうやったかな）の数々で支持の固いファンタジーゾーンもそう簡単に人気は衰えそうもない。上位3強は，来月どれがトップになってもおかしくないという今年のパ・リーグのような状態になっています（もう今となっては現実感のない表現か？）。

この3本になかなか割り込めないRPG勢ですが，先月登場の2本はどうも明暗クッキリという感じですね。ローグ・アライアンスはノーマルX1ユーザーの涙と鼻水まじりの声援を受けて

A-JAX

モトス

ディンクの**ダブルイーグル**も，ゆっくり楽しめてうれしいゲームです。

一方，アドベンチャーには「第４のユニット」の４作目に当たるデータウエストの**Zerø**が着々と進行中です。これもファンにとってはうれしい限りですね。

おっと，忘れちゃいけないビッグニュースがありました。シミュレーションの光栄から**信長の野望**と**水滸伝**が出るようです。シミュレーションの面白さをおもいっきり満喫してくださいね。あと呉ソフトウエアからもシミュレーションゲーム**ファーストクイーン**が出ます。あのゴチャキャラは見ていても楽しいものですよ。そうそう，工画堂スタジオからも**シュヴァルツシルト**がいよいよ登場します。まさにシミュレーション真っ盛りといったところでしょうか。

最後になってしまいましたが，ハドソンから**上海２**が出るようです。ひさしぶりのハドソン，楽しみにしたいですね。

大幅にランクアップを果たしましたが，リングマスター1は早くもランクダウン。サバッシュもつきあって下がってしまいましたね。あの程度が限界とも思えないし，もっと頑張ってほしいんですが。

前からいやに居残るなと思っていたソーサリアンが何故か今になって５位にアップ。追加シナリオだけではもう説明がつかない人気の長さ。いまやX1turboユーザーの心の友になっているようです。

今月初登場はブローダーバンドのWings。新手のX68000用アクションですが，このメンバーが相手では大幅なランクアップは難しいぞ。

ま，何にしてもですね，上位３強に喰い込めるソフトの登場を楽しみにしたいと思う浦川でした。ではまた来月。

新作ソフト情報

☆……11月１日現在発売中　★……近日発売予定

明記されたもの以外の価格については消費税は含まれておりません。

☆Heroes of the Lance

信仰心をなくした人々の前に復活した，暗黒の女王タキシス。彼女を倒すには，真の神と人間との交流を復活させるというミシャカルの円盤を手に入れなければならない。平和な世界を取り戻すため冒険者たちは旅立った。

ゲームはアクションゲームで構成され８人のパーティメンバーの中からその場に応じたキャラクタを選び，剣や魔法で戦っていく。

X1/turbo用　5″2D版４枚組　7,500円
ポニーキャニオン　☎03(221)3161

★倉庫番パーフェクト

かつて数多くの人々の頭を悩ました倉庫番が復活する。さらに練り上げられたステージは現在300面以上がスタンバイ。グラフィックももちろん描き直され，エディタも付いてくる。さらにはエディタを使ったコンテストも企画されているらしい。

X1/turbo用　5″2D版　6,800円
シンキングラビット　☎0797(73)3113

★V'BALL

「熱血高校ドッジボール部」に続くテクノスジャパンのビーチバレーゲーム。バレーとはいっても，選手は２人だけ。７点先取の１セットマッチだからスピーディに試合が運ぶ。プレイヤーチームはアメリカ沿岸を舞台に試合をしていく。トスはコンピュータに任せて，プレイヤーはサーブ，レシーブとアタック，それからブロックを担当。大作のように構えず手軽にできるのがうれしい。

X68000用　5″2HD版　7,900円
シャープ　☎03(260)1161

★レナム

1500年前に封印された邪教神バーディス。だが，彼は目覚めかけ，自らの復活を図り始めた。しかし，封印を行った魔法種族「レナム」は今では半ば伝説と化し，再び封印をしようにも存在しているのかさえもわからない。この「レナム」を捜すために旅に出たひとりの娘ティアがラ・エールの街に着いたところから，物語は始まる……。

豊富なセリフでキャラクタの魅力を描き出すマウス操作のRPG。街から街への移動などに独自の工夫も見られる，新鋭ソフトハウスの期待の一作。

X68000用　5″2HD版６枚組　9,800円
ヘルツエンジニアリング　☎03(371)3081

ラグーン

フラッピー２

やじうまペナントレース

シュヴァルツシルト

Heroes of the Lance

倉庫番パーフェクト

レナム

☆**クランクトアロウ**

ログインのソフトウェアコンテスト入賞作品。独特の飛行感が気持ちいいシューティングゲーム。

戦闘機クランシーの目的は，敵の輸送車カーゴを破壊すること。クランシーは一度に12発しか爆弾を持てないうえに，敵の対空車両がカーゴを守っているので，効率のいい攻撃をしなければならない。弾薬が尽きたら高度を上げ，空中給油機で補給を受ける。先の面では超巨大戦車との対決もあるというなかなかツボを押さえたゲームだ。

X68000用　5″2HD版　2,000円
ブラザー工業　☎052(824)2493

☆**Misty**

シナリオを辿ることよりも，プレイヤーに推理をさせることに主眼を置いた斬新なゲーム。プレイヤーは事件のポイントをまず頭に置いて捜査に出かける。情報を集めながら推理を進め，謎が解けた！　と思ったら質問に答える形で推理を示す。それが正しければ，事件の真相が語られるというわけだ。X68000用はシナリオに焦点を絞ったのか，グラフィックはモノクロになっている。

X68000用　5″2HD版　5,000円
データウエスト　☎06(968)1236

★**バトルチェス**

1人用と2人対戦用のほか，コンピュータ同士の対戦を観戦できるモード付きのチェスゲーム。

2次元モードと3次元モードがあり，3次元モードでは人間の姿をした駒たちがチェスボード上で動き回るというしくみだ。また通信機能を搭載しているので，電話回線による離れた場所どうしでもプレイできるというのもうれしい。

X68000用　5″2HD版2枚組　9,800円
パック・イン・ビデオ　☎03(5565)8732

★**GAMMA PLANET**

久々のワイヤーフレームの3Dシューティングゲーム。題材は都市の戦車戦。降下してくる敵戦車を撃退するという内容だ。

パワーアップパーツも用意されているし，装甲車のくせにバーニアを吹かしてジャンプもできるあたりは，さすがにいま風。シンプルながら骨太な面白さを持ったゲームだといえるだろう。

X68000のきれいなグラフィックで繰り広げられるワイヤーフレームの世界はなかなかの美しさだ。

X68000用　5″2HD版　価格未定
コムパック　☎03(375)3401

☆**大戦略マップコレクション**

コンピュータシミュレーションゲームの巨頭となった大戦略シリーズ。人気の理由は，ひとつには手軽なルールでありながら，マップや兵器構成によってプレイバリエーションがさまざまに広がる点だろう。そのバリエーションをさらに広げてくれるのが，このマップコレクションだ。ユーザーから募集し，厳選した40のマップが収められている。まったくの仮想マップから，世界地図から取ったものまで，いろんな「IF」の戦いが楽しめる。

X68000用　5″2HD版　5,000円
システムソフト　☎092(752)3902

☆**ダブルイーグル**

単にプレイを楽しむためのプレイゴルフモードと，30年間のゴルフ人生をシミュレートしたゴルフライフモードの2つが用意されたゴルフゲーム。

プレイゴルフモードでは12のコースが選べ，ゲームの感覚をつかむにはちょうどいいモード。このモードでゲームに慣れてきたら，ゴルフライフモードで世界のトップを目指そう。

X68000用　5″2HD版2枚組　9,500円
アートディンク　☎0474(77)7541

★**アルビオン**

発達した科学技術が引き起こした最終戦争から数百年，人々は神官を通じてテクノロジーを授けてくれるコンピュータを神として崇め，平穏な日日を送っていた。が，ある神官の野心が破壊神を呼び起こしてしまった。主神に見出された少年は，破壊神を倒すために旅に出るのであった。

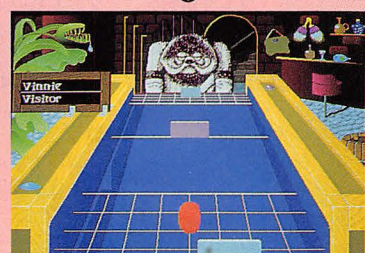
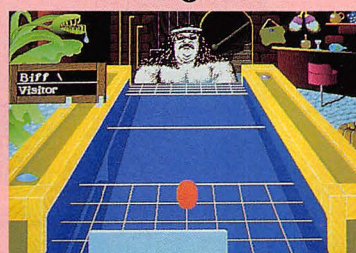
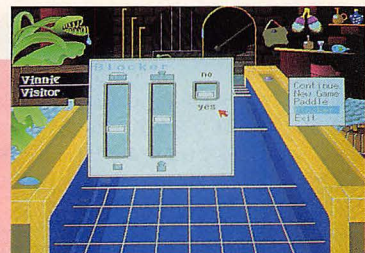
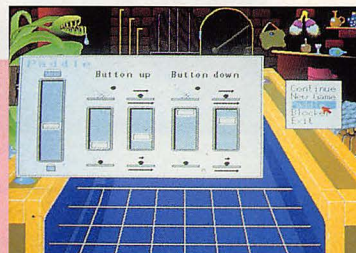
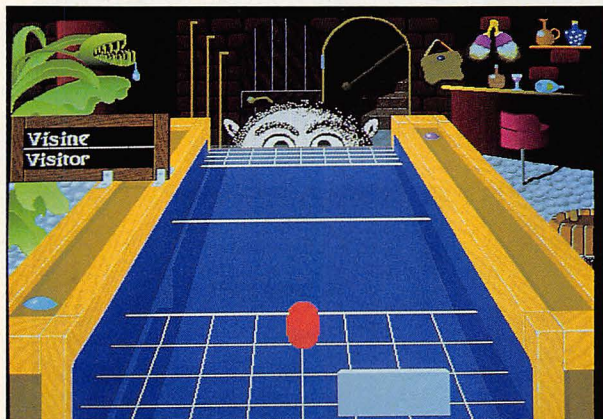
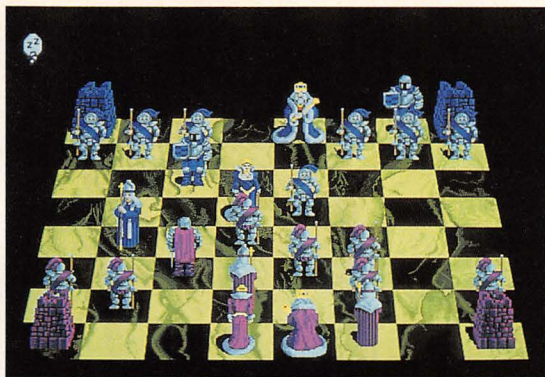
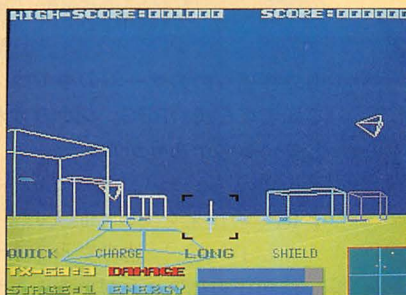
3DダンジョンタイプのRPGで，アイテムを捜しながら武器やサイキックポイントを使ってモンスターを倒し，進んでいくというゲームだ。

X68000用　5″2HD版2枚組　9,800円
カオス　☎06(927)1060

★**Zerø**

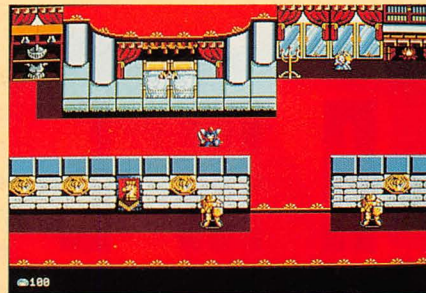
第4のユニットのVol. 4。WWWFにおける「DUAL

シャッフル・パック・カフェ

マウスをパドルに見立ててパックを打つエアホッケーゲーム。スコンスコーンという音が心地よく響き，なかなか臨場感をかもし出している。対戦相手は11人。パドルの大きさを変えたり，障害物を置いたりして楽しめる。

いまどきなかなかお目にかかれないエアホッケーをゲームにするところが斬新ですね。実際にこのゲームをやってみると意外と難しいのです。それと，ゲームを始める前に周りをかたずけておかないと，あとでとんでもない目にあいますよ。

バトルチェス

GAMMA PLANET

大戦略マップコレクション

ダブルイーグル

サイバーミッション

ファーストクイーン

TARGETS」作戦後，自分の正体を知るべく軍の情報管理局へBS関係資料閲覧願いを提出したブロンウィンだが，ひとつの疑問が持ち上がる。スイシーゼが解放した「越中優治」がWWWFのクローンではないか……と。そしてついに軍最大の司法機構「監査局」が動き出した。今回は選択肢に表情コマンドを導入した。

X68000用　5″2HD版　8,800円
データウエスト　☎06(968)1236

★メタルサイト

プレイヤーは5人の兵士から選んでスタート。入隊試験から始まって，敵軍の衛星兵器を破壊するまでさまざまなミッションをクリアしていく。が，敵キャラはすさまじいスピードで出てくるわ，ミサイルは飛びかうわでもうタイヘン。しかも画面をふさぐようなデカキャラも登場，果ては障害物だらけの通路の飛行までやらされる。ジョイスティックを持つ手が汗ばむほどのスピーディさとスリリングさは保証つき。

X68000用　5″2HD版5枚組　8,800円
システムサコム　☎03(635)5145

★ナイトアームズ

地球からはるか彼方の「かみのけ超銀河団」では，連邦軍と宿敵CIPHERの戦いが繰り広げられていた。連邦軍はCIPHERの作り出した「ステラ・スマッシャー」という強力な武器を破壊するため，追撃兵器「ナイトアームズ」を出撃させた。

前後の3Dに加え，横にも3Dスクロールするという，全方向3Dシューティングゲーム。X68000ならではのグラフィックは見応えもたっぷり。

X68000用　5″2HD版　9,700円
アルシスソフトウェア　☎0956(22)3881

☆サイバーミッション

ログインのソフトウェアコンテストで入賞したシューティングゲーム。ゲーム中に現れるグレーのカプセルを8回打ち，出てきた色つきのカプセルを自機やオプションが取ることによってパワーアップしていく，というしくみだ。また，別々にパワーアップさせた自機とオプションを合体させると，より強力な攻撃ができるようになる。

X68000用　5″2HD版　2,000円
ブラザー工業　☎052(824)2493

★ファーストクイーン

X1に発売されていた“シルバーゴースト”の延長線上に当たるゲーム。オルニック女王の治めているログリス王国再統一の野望を阻止するのが目的。プレイヤーは軍の編隊を組んだのち，リーダーとなるキャラクタに命令を与えて操作する。戦闘シーンになると最大100人のキャラが入り乱れて戦うので画面はもう大騒ぎってカンジだ。

X68000用　5″2HD版　価格未定
呉ソフトウエア　☎0486(46)0660

★た～みのる2

X68000用のパソコン通信ソフト「た～みのる」がバージョンアップされた。旧バージョンより使いやすくなったので，パソコン通信の初心者にはうってつけってところでしょう。詳しくは72ページを参照のこと。

X68000用　5″2HD版　17,800円
エス・ピー・エス　☎0245(45)5777

★サイバーノート

住所録の整理や名刺管理というのは意外と面倒なもの。このサイバーノートは，そんなわずらわしい作業をマウス操作で簡単にできるパーソナルデータベースだ。機能としては，スケジュール機能，カレンダー機能，家計簿管理機能，住所録&名刺管理機能などがついている。もちろんシャープの電子手帳とのデータ交換や，ほかのソフトとのデータコンバートが可能というスグレモノだ。

X68000用　5″2HD版　19,800円
シャープ　☎03(260)1161

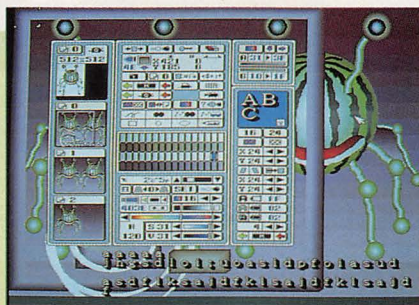
★マジックパレット

65536色より256色の画面を4枚メモリ上に作成でき，それぞれのセーブ/ロード/コピーできるという超高速グラフィックエディタ。コピーやペーストの機能も充実，描画テクニックとして活用できるユニークなアンドゥ機能もついている。開発用にも最適。

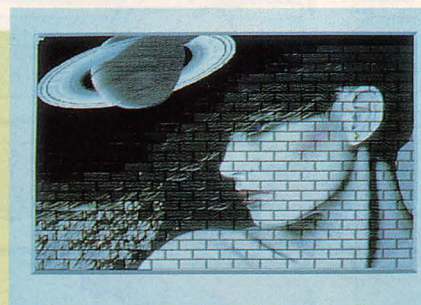
X68000用　5″2HD版　19,800円
ミュージカル・プラン　☎03(401)2751

★G68K versionⅡ-PRO

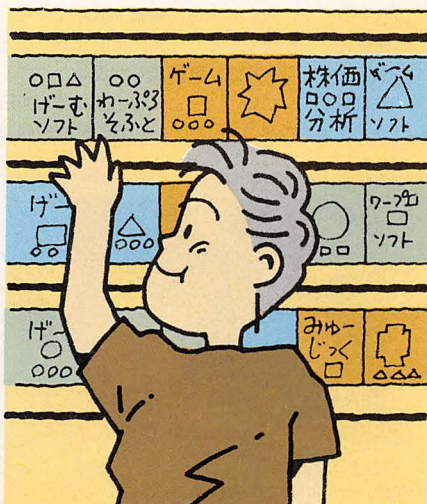
すでに発売されているグラフィックツールG68Kのバージョンアップ版ができた。この製品は，前のバージョンと同じく操作が簡単でありながら，処理スピードが速くなったうえ，エアブラシなどの機能もアップ，高速，高機能を実現した。

X68000用　5″2HD版　22,000円
システムハウスオー　☎075(822)4408

マジックパレット

G68K versionⅡ-PROの作品

THE SOFTOUCH

GAME REVIEW

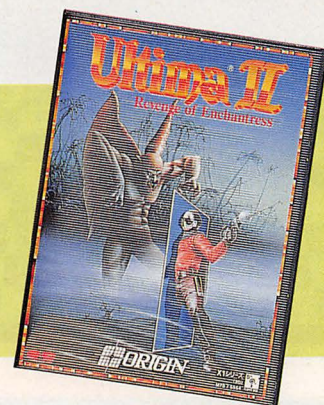
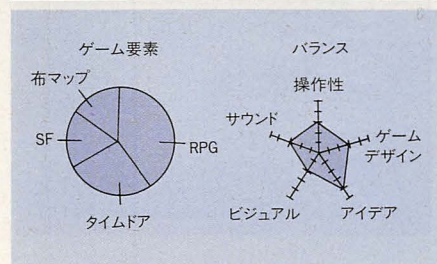
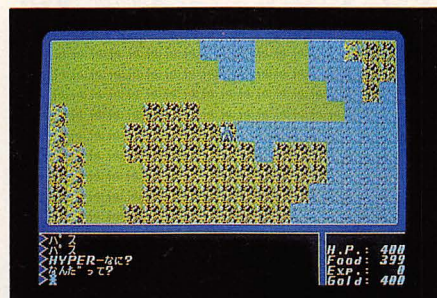
GAME REVIEW

今月もいろいろなジャンルのゲームが出揃いました。まず，X1用にはRPGの大御所とも言える「UltimaⅡ」を，そしてX68000にはテキストアドベンチャーゲーム「Misty」と，ちょっと難易度が高いシューティングゲーム「メタルサイト」を用意しました。

UltimaⅡ

海外移植もののゲームの中では，五指に入る知名度を持つRPG。シナリオがしっかりしているのがありがたい。

▶RPGの出来を決める最大の要素はシナリオでしょうが，一連のウルティマシリーズには熱狂的なファンがいることからも，このゲームのシナリオの完成度の高さはみんなが認めるところ。今回は時間を往来することができるタイムドアが出てきて，過去から未来まで5つの時代の冒険が楽しめる。さらには宇宙にだって行けちゃうというんだから，どんな魅力的なクエストが用意されているのか気になるところだ。しかし，だ。実際に遊んでみたら，キーバッファに悩まされることはなくなったけど，相変わらずゲームスピードが遅くてスクロールがタコなんだよな。某氏がこれを見て「X1でラスタースクロールをやっている」(画面が波打つさま）と言っちゃうくらいなんですから……。それでも一見する価値はあるかもしれません。せっかくPCGがあるんだからX1の機能を生かしてほしかった。これではゲームの世界にハマリこむ前に，飽きてしまうことにもなりかねないもの。

熱中度▶▶▶▷▷▷▷　(H.K.)

▶ウルティマの2作目。古い！　なんて言うなかれ。馬に乗ったり，船に乗ったり，飛行機を操縦したり，あげくの果てには宇宙船で宇宙に旅に出る！　っていうんだからもの凄い発想です。前作で，悪の魔導師モンデインは打倒されました。今回は，その弟子のミナクスを退治するというシナリオです。もちろん，またもやロード・ブリティッシュが登場するようです。

基本的に操作法やら，船を奪って大砲を打ちまくるところやらは同じなので，Ultimaファンにはなじみやすいでしょう。でも，もっと驚いちゃうのは，タイムドアの存在です。これを使えば，大昔から21世紀まで好きな時代に行かれるんですが，伝説の時代には「伝説の時代」と書かれた看板が立っているので笑えます。なんにしても，「時空を越えた究極の冒険」というコピーもまんざらウソではないでしょう。時間を越え，宇宙を探険するアメリカ人のSF感覚には脱帽します。

熱中度▶▶▶▶▷▷▷　(澤)

X1turbo用　5″2D版3枚組　7,800円(税別)
ポニーキャニオン　☎03(221)3151

Misty

「じっくりと考え，楽しんでいただくためグラフィックや音楽をあえて排除した」というテキストアドベンチャーゲーム。

▶まず，初めに言っておくが，この「Misty」はテキストアドベンチャーゲームですぜ。っていうことは絵がほとんどない。だから，FM TOWNS版の派手なデモを見てその印象が強い人は別のゲームであると考えたほうがいいですぜ，ダンナ。

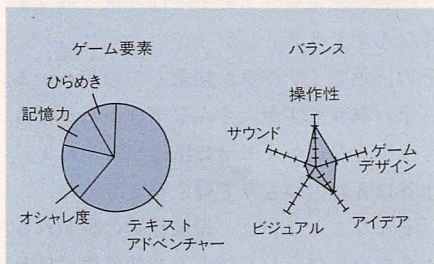
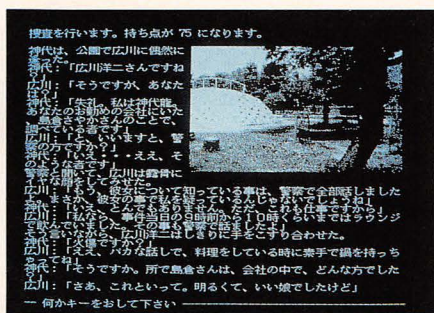
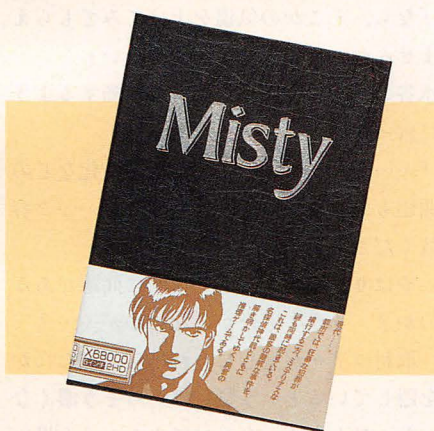
このゲーム，5つのシナリオが入っていてどれもがショート。リバーヒルソフトの「J. B.シリーズ」の聞き込み中心の捜査とは趣向が違い，クイズのようにいかに少ない情報で事件の真相を把握できるか，というようなゲーム内容だ。

さて，シナリオの内容だが，かなり事件内容は現実に近く設定されている。シナリオ1と5はほとんどヒントなしでも事件が見え見えで初心者向き，一方，2，3，4はかなり難しく上級者向きといえる。8ビットパソコン（X1turboなど）にも移植されるそうなので，最近のチャラチャラした

グラフィックアドベンチャーに飽きた人はどうぞ。
熱中度▶▶▶▶▷▷▷ (善)

▶5つのストーリーが詰まった，探偵もののアドベンチャーです。っていうよりミニ推理小説とでも呼んだほうが適当かな。なんせ基本はテキストをだらだらと読むだけ。ときどき申し訳程度にグラフィックが現れるのだけど，なんとモノクロ。おまけにBGMもなしという，X68000の高性能を見事に無視してくれています。

で，勝負はストーリーの良し悪しにかかってくるんだけど，まあ普通とでも申しておきましょうか。だいたい私は，入り組んだ人間関係がどーとかこーとかいう殺人事件の類はあまり好きじゃないんですね。

だけど，データウエストとしては，ユーザー参加による盛り上がりに期待を寄せているんじゃないかな。以降発売されていく（らしい）Mistyの続編のシナリオを募集し，採用の際には10万円の賞金を出すということですから。腕に自信のある人は，どうぞがんばって5,000円で10万円を釣ってくださいませ。
熱中度▶▶▷▷▷▷▷ (お)

X68000用　5″2HD版　5,000円(税別)
データウエスト　☎06(968)1236

メタルサイト

少々難易度が高い3Dタイプのシューティングゲーム。操作性もよく画面もきれいなので，じっくり腰をすえてプレイしたい。

▶期待の新作，システムサコムのメタルサイトが登場です。

3Dタイプのオリジナルシューティングゲームなんですが，素晴らしい出来ばえになっています。なんといってもこのゲームの最大の目玉はスムーズな3D処理でしょう。しかも高速で画面の半分もあるようなデカいキャラクタが2つも同時に動き回ってしまうんですから，サコムが独自開発した疑似スプライトシステムの威力がわかります。

ゲームは全10面で構成されていて，最終目的は敵の宇宙要塞にある大型兵器を破壊することなんですが，それぞれ面ごとのバリエーションが豊富で，特に7面のカミナリが超ハデで見ていて気持ちがいいほどです。やられたときの自機の爆発も画面いっぱいに，これまたいやというほどハデに爆発してくれます。ただ敵の攻撃がもの凄く一般人の私にはとてもついていけません。ハードゲーマーにおすすめの1本でしょう。
熱中度▶▶▶▶▶▷▷ (純)

ゲーム要素　バランス
ハデさ　マニア度　シューティング
操作性　サウンド　ゲームデザイン　ビジュアル　アイデア

▶フッフッフッフッフ。私はこの手のゲームを待ち望んでいたのだ。大抵のゲームではいきなり私は○○軍のエースパイロットであり，地球の運命をかけて××軍の本拠地へ単機突入！　などという，理不尽極まりない設定で，一兵卒だったころの私，もしくは士官学校時代の私などはどこにもいない。入隊して（ゲームを買って）いきなりトップでは○○軍のレベルの低さがわかろうというもの。これじゃあ地球が滅びても文句は言えない。いくら天才パイロットとはいえ下積み時代はあるもんだ。このゲームではまず入隊テストがある。つまり私は自衛官募集の広告につられて志願したのである。そして百戦練磨，一騎当千のエースパイロットへと特進するのだ！　ただ，悲しいかな最前線中の最前線らしく，指令は届いても武器，弾薬は届かない。なぜか修理もできない。敵は圧倒的物量作戦できているのに，である。せめて補給は欲しかった。ちょっとムズイぞ。
熱中度▶▶▶▶▶▷▷ (S.K.)

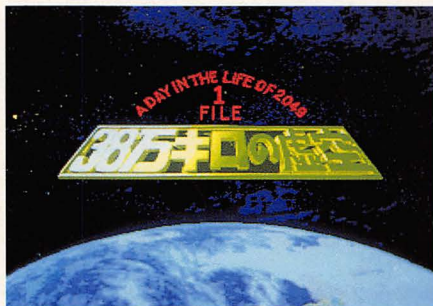
X68000用　5″2HD版2枚組　9,800円(税別)
システムサコム　☎03(635)7609

ダメなら自分で作るけど

今月も編集部にアドベンチャーゲームがやってきました。「38万キロの虚空」と「Misty」がそうなんですが，この2つ，恐るべき共通点があります。そう，それはどちらも文字が異常に多いのです。ふだん「第4のユニット」など比較的文字の少ないアドベンチャーに慣れていた私は，大パニックに陥ってしまいました。

まあ，アドベンチャーっていうのは文字が多いものなんだろうけど，やっぱり人に長い文章を読ませるんだったら，もうちょっと工夫がほしいです。これじゃちょっと目が疲れてしまうんですよね。で，さっき編集室で話してたのが「シャープワープロ式メッセージ出力」。アドベンチャーゲームのメッセージウィンドウっていうのは，たいてい文字の出るスピードの調整かページ切り替えで，次のページに送っていただけでしょ？　そうじゃなくって，ワープロみたいにマウスでテキストを前後どっちにもスクロールできるようにして，プレイヤーが自分でメッセージを動かせるようにするんですよ。そうすれば，自分の好きなスピードでメッセージを読むことができますし，読み直しもきくわけです。ひとつやってみてくれませんか，ソフトハウスさま。 (で)

THE SOFTOUCH

●38万キロの虚空

宇宙の果てに人間ドラマを見た

komura Satoshi
古村 聡

システムサコムお得意のノベルウェアシリーズ第5弾。近未来のスペースコロニーを舞台にしたアドベンチャーゲーム。こたつに入ってじっくりと楽しみたい，といった感じのゲームです。

X68000用　5"2HD版4枚組　9,800円(税別)
システムサコム　☎03(635)7609

事件への第一歩

俺の名は相場謙。ONNの社会部記事担当の事件記者だ。いま，ONNの同僚ドワイト，ステファンとともにスペースコロニー1，通称SC1にいる。いや，ONNだけではなくABCやCBSといった大手ネットワークの連中やその他大勢もごちゃまんとだ。なぜなら大々的なオープニングセレモニーを3日後に控えて，多数のプレスの記者たちがこのSC1へやってきたのだ。

2049年。人類はついに宇宙への移民を開始することになった。SC1は初めて建造された大規模なコロニーの第1号だ。オープニングセレモニーにはこのコロニーを建造したNSCA会長クロフォードも出席し，その様子は全世界に中継されることになっている。

ここ，SC1に入るときに手荷物の検査があったが，銃を持った武装警備員がずらっと並んでいたのにはびっくりした。そして，全員の荷物をかたっぱしからそこいらじゅうに広げ，荷物を1つひとつ調べていくというとんでもなく厳重な検査をされた。まったくなんて連中なんだ，NSCAのやつらは。あの態度は頭にくる。ちょっと警備員に話しかけようとしたら銃を突き付けられるわ，広報担当官にはほとんど脅迫のような注意は受けるわで，もうさんざんだ。おまけにNSCA報道担当主任のイザベラ・スローターときたら緊張してるのかえらく冷たいロボット女みたいだ。それに最後に配られた腕時計状の機械。IDカードのようなものだという説明だったが，どうもわれわれの動きを見張るトレーサーのような気がする……。まるで俺たちが報道関係者というよりは敵国人のような扱いだ。なにかNSCAにはプレスは敵だとみなさなければならないようなことでもあるのだろうか。

SC1ハイジャックさる!

「このコロニーは乗っ取られている
差出人不明」

なんだ!?　タレコミか?　しかも，差し出し人不明とは……?　通常，メイルは差し出し人不明などという芸当はどうやっても不可能なはずなのだ。俺専用の情報屋，ジャックでさえも不可能な芸当なのだ。

コロニーが乗っ取られている……。本当だとしたらたいへんな事件だ。しかしどうやって乗っ取ったというのだ?　とにかく本当か嘘かどっちにしても確証を得たい。なんとか裏を取らなければ……。

取材陣はNSCAの広報官の案内でコロニーの内部を取材することになっていた。いわばNSCA主催のコロニー見学ツアーというわけだ。まずは天気や気温を制御するコントロールルームに案内された。これはなにか探れるぞ，と思った俺は昨日の気温がかなり高かったことについて質問してみた。

「えっ，気温ですか……。実は最終テストを兼ねて少し高めに設定しているのです」

質問に答えたイザベラはかなり動揺しているようだ。

「ほう，作業中の人たちもいるのにですか」

AP通信のジェームズ水谷が言った。

「皆さんのご心配はわかりますがここの環境コントロールについては万全です。なにしろここには調整も取り換えもきく太陽がついているようなものですから」

「なら，どこかの気温を上げてみてもらえませんか。目で見るのが一番いい」

水谷の言葉に彼女は狼狽して懇願するように言った。

「……そ，それは……。環境の変化などの問題もありますので，残念ですが……ご容赦ください」

やはり，環境コントロールに問題があるのか?　それともほかのなにかが……?

取材ツアーが進むにつれNSCAがなにかを隠しているという疑いはいっそう濃くなった。ひとつカマをかけてみるか，と思い，わざとイザベラに聞こえるような声で暇そうにしている水谷に話しかけた。

「平和そのものだけど，これでけっこう裏で事件が起きてたりしてね」

「事件ですか?　相場さんお得意のジャンルですね。でも，いま起きる事件ってどんなもんですかね?」

うまい具合に水谷は話にのってきた。

「一発特ダネクラスでやっぱり乗っ取り事件なんかどう?」

その言葉でイザベラが緊張したのがわかる。

「乗っ取りですか。そんな事件があったらもう少しなにかオモテに出ませんかねぇ」

水谷は冗談のつもりで軽く答えてきた。

「そんなことないよ。報道管制の敷かれるような事件だったらほとんど部外者にはわ

からないもんさ」

イザベラが我慢しきれなくなったのかこっちを向いた。よしよし，こっちへこい！

「随分と物騒なお話ですわね？」

やはり，彼女はさっきから話を聞いていたことになる。と，いうことは！

「いや，ちょっとした空想ですけどね。どうです？

“SC1，ハイジャックさる!?”

なんて見出し，間違いなくウケますよ」

「それじゃ，誤報になっちゃいますよ」

「だから“!?”をつけてね。仮想事件ってヤツですよ。どうすればSCを乗っ取れるか考えてみるのも面白いと思いません？」

「あまり，お薦めはしませんわ。そんな記事をお書きになりますと取材許可が取り消されるかもしれませんよ」

彼女の口調はほとんど脅迫だった。

「実際，乗っ取られるわけないんだからそんな真剣にならなくても……。それとも？」

「まさか，そんなことありえません」

「でしょうね，もし強制退去なんてことになったらそれこそ疑いますよ」

彼女はかなり緊張したらしく冷や汗さえかいているように見える。間違いなく乗っ取りはあり，しかもNSCAは相当危険な状況にある……俺はそう確信した。

強制送還・かけ引き

俺は宿舎に帰って同僚のドワイトとステファンの2人にハイジャックのタレコミについてとイザベラの態度について話した。

「差し出し人不明のタレコミとはおそれいったな」

ドワイトは言った。

「確かに昨日の気温についても怪しいことだらけだしな……」

ステファンはそのタレコミが真実であると確信していたようだ。なにか先に情報をつかんでいたのだろうか？

まあいい。俺にも考えがあった。俺は記者だ。だったらこの乗っ取りの話を噂として記事に盛り込んでしまえばいい。かなりきわどい内容になってNSCAから目を付けられるかもしれないが，イザベラには釘を刺したし，うまくいけば世論が盛り上がってSC廃止へと持っていけるかもしれない。俺は一気にその記事を書き上げ，ベッドに倒れ込んだ。疲れのせいかすぐに眠れた。

「謙！　おまえなんてもん書いたんだ！」

次の朝はドワイトの罵声で目が覚めた。

「おまえの，あの記事のおかげで即刻，強制送還になっちまったぞ！」

「え!?」

「わからんのか？　NSCAの圧力だよ！乗っ取りの噂なんか立てるから，名誉毀損で訴えるって話になってるらしいぞ！」

「で，記事は差し止めか？」

「ああ，誤報ってヤツだ」

ステファンは苦々しげに言った。彼もそれが誤報ではないことを知っているのだ。

数分後，犯罪人の護送車のようなNSCAの車がきた。まったくなんてことだ……。

「大丈夫だ，2人とも。地球なんかに戻されやしないさ」

「戻されない？」

「宿舎の中じゃ盗聴されているだろうから言わなかったんだが……。もう遅いんだよ」

「どういうことだ，ステファン」

「行けばわかるさ……」

15分後，俺たちを乗せた車がシャフトステーションで止まると，そこには例のイザベラの冷たい視線があった。

「このような形になってしまい遺憾の極みです。30分後のシャトルで地球に戻っていただきます」

彼女は淡々と告げた。

「ひと言……言わせてもらっていいかな」

ステファンはゆっくりと口を開いた。彼はなにをするつもりなのか？

オープニングデモ画面

最後に

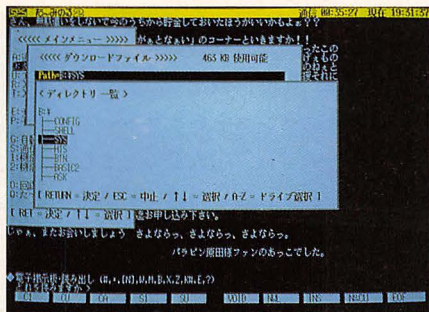
という具合に話は進んでいきます。さてこのゲーム，システムサコムのノベルウェアシリーズの5作目で，DOME 系列の近未来SF小説という形をとっています。主人公は相場謙というネットワークONNの日本人記者，舞台は2049年の地球，そして初の大規模スペースコロニーSC1。それをとりまく役人，記者，政治家，テロリストたちを巡るサスペンスというストーリーです。

NSCA報道担当主任イザベラ・スローター

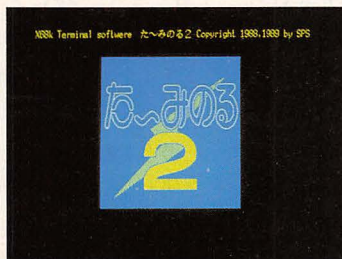
このゲームでは，主人公のとる行動によってストーリーに若干の違いが出てきます。いまここまで書いたものは，私のやったシナリオの中で（全部で7通りぐらいやったかなあ）一番スリリングで面白かったものを元にしています。おそらくこのゲームの一番の売りであるこのシステムのおかげで，人によってシナリオに対する評価はかなり分かれてしまうでしょう。一番スリリングなパターンでは，このようにカマをかけてみたり強制送還にあったりとイベントが多くてかなり楽しめるのですが，他のおとなしいパターンではイベントがあまり過激なものではなく，どうもストーリーのメリハリに欠けるきらいがあるように思えます。

さて，システムの話ですが，これはかなり頑張っているように思います。実はこのゲーム，MIDI対応になっているのですがこれははっきり言って凄い！　音色は，MT-32に対応なのですが，一応ほかの楽器でも試そうかと，D-10なんかも使ってみました。で，結果はというと，もう最高。SF文庫本がいきなり映画館に早変わりってぐらいに凄いのです。ぜひ部屋を真っ暗にして MIDIを使って音を聞いてみてください。

ただ，全体としてみたら手放しで喜べない部分もけっこうあります。特にメッセージウィンドウなどがそうで「ソフトでハードな物語2」ではウィンドウに似顔絵があったのになぜこれではなくしたのかとか，やっぱりメッセージ出力が遅いとか，なんで768×512モードを使わないのかなどの問題がそうです。シナリオなどの面ではかなりレベルが上がってきているので，今度はメッセージ周りのシステムのフルモデルチェンジをお願いしたいものです。

●た〜みのる2

パソ通入門者にも安心の通信ソフト

Fukuhara Toru
福原　徹

X68000で代表的な通信ソフトのひとつである「た〜みのる」がバージョンアップされました。ビジネスからユーザー同士のコミュケーションまで，多様な可能性を持つパソコン通信をこの1本でサポート。

X68000用　5"2HD版　17,800円(税別)
エス・ピー・エス　☎0245(45)5777

発表当初から，その“変な”名前で話題になっていた，SPS開発の通信ソフト「た〜みのる」が「2」になって再び僕らの前に現れました。名前だけ聞くとなにやらゲームのようにも思えますが，これは紛れもなく通信ソフトです。事実，私はこれのバージョン1で日々アクセスを営んでいたのでした。

パソ通の楽しみ

ど〜も「かわはらゆい」です……と言って私が何者だかわかる人がいたら，あなたはパソ通のベテランです。えっ，誰もわからんって？　失礼しました。

さて，パソコン通信といえば，近ごろではコンピュータ関連雑誌のHOW TO記事にはもちろんのこと，某大手電気メーカー提供のミーハードラマ[1]のネタにされてしまったり，天下の読売新聞社がホストを開局してしまったりと，話題提供には事欠きませんね。

一説によると，日本の個人ネットワーカーは約10万人，企業を含めても30万程度だそうです（あまり信用しないように。正確には覚えていない)。ただ，この数字を多いと見るか少ないと見るかはあなたの自由ですが，パソコン人口から比較して，十分発達浸透したメディアだとは決して言えないと思います。私の経験からいくと，ネットワーカーには，パソコンマニアよりもむしろ，常に新しい情報を必要としているマスコミ関係者，ビジネスマンが多いような気がします。

ただし，難しく考えてはいけないんですね。1回アクセスを経験してみればわかることですけど，ネットには多種多用な趣味，仕事を持った人が大勢入ってきます。年齢だってさまざまです。ですからコンピュータの話題だけではなく，芸術・音楽，文化，医療，時事・社会問題，バイク，もちろんアニメやSF映画の話題もありますよ。それぞれが自分と波長の合った仲間を見つけてアクティブに活動しています[2]。

そうはいっても根がケチな私は，あまり大きな声でネットの楽しさを宣伝したくない気持ちも若干あるのです。少数精鋭の現在のネットの状況が，非常に居心地がいいというのもあるし，某朝ドラ[3]に影響されて，女の子目当てでやってくる（女の子なんているわきゃね〜だろ）ナンパ野郎が増えるのも気分が悪いですから。

おっと，今回はパソ通の解説じゃなかったんですね。いいかげんに本題に入りましょうか。

これだけ変わった

さっそく，前バージョンより追加，変更された機能のうち，目立った点を書いてみましょう。

●通信速度　300〜19200bps対応

旧バージョンの最高9600bpsから，19200bpsにアップ。これはほかの通信手段(直接マシン同士をつないだ場合のデータ通信など）の場合に有効だ。公衆回線を使った普通のパソコン通信の場合には，現在一般的に使われているホストが速くても2400bps程度なのであまり関係ない。

●16進表示による受信文字表示

仕様の異なるモデム間でコマンドのやりとりを行う場合などに，実際にどんな手順でデータがきているのかを見ることができる。また，ホストによっては特殊なコードを送ってくるところもあるため，おかしいなと思ったら16進で調べられる。

●オリジナルエディタの搭載

オンラインで文書をエディットする場合などに使える（私はそんな恐ろしいことはしないが……)。

●画面表示色の設定変更

背景色を変えたり，VT-100に対応しているホストとの通信のとき，文字の色や属性を自分の好みに変えられる。

●画面モードのリアルタイム変更

旧タイプでは，24kHzのディスプレイが必要であったが，た〜みのる2では31kHzオンリーのディスプレイでも使えるように改良されている。これでCZ-603Dユーザーも安心して使用できる（ただし31kHzでの使用時は横96カラムモードオンリーね)。もちろん24kHzのディスプレイなら，80カラムでの使用が可能。

●カラムゲージ表示

画面の上にカラムゲージを表示している。邪魔だと思うなら消去可能だ。

●チャット用1ラインエディタ

日本語フロントプロセッサと，電話回線の間に1ラインのバッファが付いたと考えればわかりやすい。旧バージョンではチャット中，フロントプロセッサでの変換に手間取ったりすると，変な変換をしたまま送信してしまったりしたものだが，1ラインのバッファが付いたことで，内容をよく吟味してから一括で送信することができる。チャット中でなくても使えるので，オンライン書き込みをよくする人にも便利。

●ファンクションキー・ユーザーキーの設定

ファンクションキーは前バージョンでもユーザーに開放されていたが，2ではA

～Ｚまでのキーにマクロも登録できるようになった。[SHIFT]＋[CTRL]でマクロモード，あとはＡ～Ｚを叩けば送信される。

●通信終了時のバックログ自動または指定保存

通信を終了すると，カレントドライブにバックログが自動的に保存されるというもの。ただ，た～みのるとHuman68kとの間を行き来していると，カレントがどこかわからなくなることもあるので注意が必要。

……と，こんなものでしょうか？

個人的には，チャット用１ラインエディタが一番おいしい機能ですね。これで変な変換（たとえば「工藤静香」と書こうとして「駆動静か」になってしまったり……）をしてハジをかくことも少なくなるわけです。本当に欲しかった機能ですよ（あって当たり前という話もある？）。

ともかく，軟弱な名前に反してとりあえず普通の通信に必要な機能はすべて備えているので，初心者にはこれだけで十分といえます。加えてX68000では当然ともいえる機能，回線をつないだままHuman68kを呼んで，メモリの許す限りの子プロセスを起動し，再び通信画面に復活するといった芸当も前バージョンそのまま。極端な話，ゲームさえできるのです（チャット中こいつで98ユーザーを驚かした）。これは使い方によっては強力な機能ですよ。

使ってみる

さて，試しにひとつ，立ち上げてみるとしましょうか。意味不明のカミナリマークオープニングがピカピカっと現れ，その後，通信画面になります。旧バージョンでは，ここでディスプレイが「カチッ」という音とともに24kHzになるのですが，２では31kHzのままです。

画面構成はすっかり変わって，旧た～みのるとは，まったく別物の様相。メニューウィンドウが白くなってしまい，右上に「現在の時刻」と「モデムの電源オンからの時間」が表示されています（できればオフフックからの時間を表示してほしかった）。あとカラムバーもあります。あれ，なにか表示されていますね。なになに「モデムの電源が切れています」ですって？　こりゃどうもご丁寧に(わかってますって)。で，モデムの電源をオン。

メインメニューの表示

私の場合，旧た～みのる用の自動ログインファイルがすでにいくつか作ってあるので今回はそれを使ってアクセスしてみました(AUTファイルはＡ：にあらかじめコピーしておいた)。メニューを開き，「Ａ」を選択します。するとAUTファイルの一覧が表示されますから，カーソルキーで選んで[RETURN]。ところがなんとOSエラー。なるほど，旧バージョンのAUTファイルは使えないのですね（ちくしょ～めんどいな～)。しかたなくHumanに戻ってエディタで修正します。

旧AUTファイルと変わったところは，

●RECIVEコマンドがRECEIVEと英語に忠実になった点（笑・英語勉強したねＳＰＳさん！）。

●モデムに対するコマンド送信がMSENDになった（ホストに対する送信は，従来どおりSEND)。

●PRINTコマンドが追加された(これが結構便利。AUTを実行中に画面にコメントを表示させることができる)。

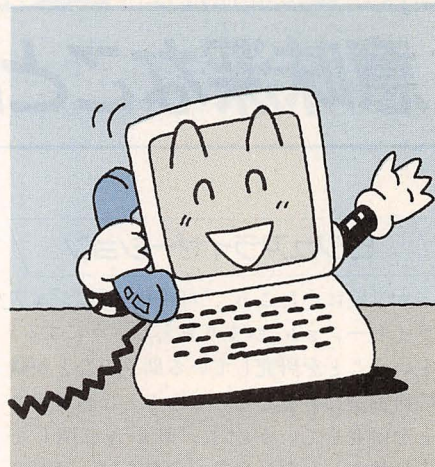
●WAITコマンドの単位が，1secから100msecになった(細かく設定できるが，25.5秒以上の待ちができなくなってしまった。これは不便かもしれない)。

●SETUPコマンドが個々に分かれてわかりやすくなった(従来N81XN，1200などと標記していた通信パラメータを，SPEED＝1200, PARITY＝NONE, DATABIT＝８，STOPBIT＝１……とそれぞれ別に設定できる）など。

で，編集を終えたら気分を入れ替えてもう一度アクセスしてみましょう。モデムがカチカチカチ（うちはプッシュホンじゃないのよ）とダイヤルしている音が聞こえて，今度こそ成功！

あとはBBSなり，チャットなり，PDSのXMODEM送受信なり，お好みでどうぞ，というわけですね。では私はPDDに画像データでも拾いにいきますか……。

より使いやすい通信環境を

というわけで，今回はた～みのる２を紹介してきました。正直なところ，使い勝手の点でまだまだ考えるべき事も多いと思いますが（いや，これはた～みのるに限ったことではない。パソコン通信自体が新しいものなので，ターミナルソフトもまだ発展途上であるといえる)，全体的にみると初心者にも中級者以上の人にもより使いやすくなったのではと思います。あと，た～みのるでは，メニュー選択のウィンドウを表示するためのキーが[HELP]に割り当てられています（[HELP] キーはX68000の場合，キーボードの端っこのほうにあるので，頻繁に使う場合は結構面倒臭い）が，それも今回は[OPT.1]キーでも使えるようになりました。できるなら両手のホームポジション内に設定してほしかったですが。それからマウスがすっかり遊んじゃっているのももったいない気がします。

通信の目的は人それぞれですが，基本的なオペレーションは，初心者もベテランもそれほど変わることはありません。通信ソフトにまず要求されることは，「多機能・高機能」よりも「あつかいやすさ」だと思います。なにしろ初心者にはちんぷんかんぷんが多い世界です（１カ月もやれば慣れますけどね)。下手に高機能なソフトに手を出して振り回されるより，最初は中程度のソフトでじっくり試してみるのがいいでしょう。この「た～みのる２」なら，これからパソコン通信を始めたいという方にもお勧めですし，ある程度通信に慣れた中級・上級者まで，十分なネットワーキングが可能でしょう。

1）　この番組を見たあと，ネットワーク上のボード書き込みの話題は「非現実的なドラマでしたね」というので埋まった（？)。沖縄からホイホイ電話をかけたり，24時間チャットモードで電話回線をつないだままにしておくなど，NTTが泣いて喜ぶような展開を見せた。

2）　気の合う仲間が見つかると，はっきりいって中毒になる。知り合いにも，NTTから月13万円の請求書を貰ったというとんでもない方がいらっしゃる。ちなみにネットワーカーはNTTからの請求書を「ラブレター」と呼び，恐れる。

3）　秋に終わったＮＨＫ朝の連続テレビドラマのこと。主役の弟はパソコン通信が趣味という設定で，よく女の子とチャットをしていた。

試用協力・みっきんネット（試験運用中・文京区）

意味深なことば「パラダイム」

ビジュアライゼーション

大学院にいたころ，「可視化」（ビジュアライゼーション，つまり見えるようにする）ということを研究している助手のひとが隣の研究室にいました。もともとは聴覚のことを研究していました。「耳が音に関してその強さしかわからないのならば，耳が2つあっても，音源の位置に関しては，頭を頂点とする円錐面上のどこかということしか特定できないはずなのに，人はなぜ音のする方向がわかるのだろう？」ということが基本的な問題だったと思います。たとえば，頭の真上5メートルのところで発生した音と頭の前5メートルのところで発生した音とを区別できるのはなぜかということです。

音や聴覚に関するこのような問題を研究していたのですが，そのうち，音の可視化を始めました。シャープのX1を買ってきてスーパーインポーズ機能などを利用したりしていました。音という空気の振動に関する情報を視覚的な情報としてディスプレイ上にうまく表現しようということです。さらにその後，匂いの可視化ということまでやり始めていました。その研究についてはどういうことをやっていたのか僕にはほとんどわかりません。

前にいた研究室も画像処理は活発でしたので，よく見る外人の女の人の顔（標準パターンらしい）を始めとして，いろいろなパターンが研究室では表示されていました。そうして，その助手の人につられて，「プログラムの可視化」ということを一瞬考えたものでした。これはもちろんプログラムの文字列自体を見るということではなくて，何らかの処理を行って，抽象的なパターンや色として一瞬のうちに何かを表現するということです。

たとえば，PASCAL的なインデントがきれいにされたプログラムがよいプログラムであるとして評価したい場合，何らかのフィルタを通せば，お行儀がよいプログラムかどうかがひと目見てわかるようにディスプレイに表示させることができるのではないかと思ったのでした。そしてそれに適当な色彩を割り付ければ，「真っ赤ないいプログラムだなあ」とか，一瞬でわかってしまうわけです。

これはまあ半分冗談ですが，目に見えるようにすることは，多くの場面で役立つ手法，あるいは考え方ではないかと思います。少し長くなりましたが，このビジュアライゼーションはひとつの「パラダイム」といえるのだそうです。

トーマス・クーンのパラダイム

パラダイム（paradigm）ということばは，もともと単に「品詞の語形変化」という意味として使われるようです。パラダイムということばが，「概念の枠組」というような意味で定義され，使われるようになったのは，トーマス・クーン[1]が1962年に書いた「科学革命の構造」という本によってでした。

この本はセンセーショナルな話題を呼び起こしたのですが，それは単にこの1冊だけの力でというわけでないようです。そのころ起きていた物理学における大きな思想的なうねりを象徴したものといえるようです。要するに，近代科学というものが，人間からまったく独立した「事実」というものを相手にしていたにもかかわらず，実はそこにおいても，観測者たる人間が介在していたという事実の「発見」です。

ひとつのパラダイムが崩れ，新しいパラダイムに移ること（パラダイムシフト）は，ことばでは表せないほどたいへんなことです。とはいいつつも，パラダイムシフトの過程を次に挙げておきます[2]。

1)　一般に受け入れているパラダイムでは説明できない異常の発見。無視したり，つじつまが合うように拡大解釈する。

2)　つじつま合わせや無視では抑え切れないほどの変則性の観測。パラダイムの誤りが認識される。

3)　新発見を説明することができる新たなパラダイムの成立。

4)　旧パラダイム側との戦い。

5)　新パラダイムが新しい発見までも予測可能ということにより受容される。

天動説から地動説という有名な話は，まさにパラダイムシフトといえます。そして，先に述べた近代的科学観というものの崩壊（ゆらぎといったほうが適切か？）もまさにそれにあたるのだといわゆるニューサイエンスではいわれてます。そして，その波は今多くのジャンルに広がりつつあるのだと主張するのです。参考までに紹介しますが，ニューサイエンスの流れは次の3つにまとめられるのだそうです。

1)　現代物理学と東洋思想との類似性の強調。

2)　自然界を支配する一般原理たる包括理論の提唱。

3)　神秘主義的アプローチ。

計算機にパラダイムはあるか？

計算機のほうの世界でパラダイムというと，まずイメージするのは，プログラミング言語の分類に使われる場面です。また，新しいプログラミング言語を売り込もうという意図のために使われることもあるでしょう。いずれにせよ，このようなパラダイムは「プログラミング・パラダイム」と呼ばれます。

プログラミング・パラダイムということばが背表紙に入った情報処理学会誌の特集[3]がここにあります。ちょっと古いのですが，「新しいプログラミング・パラダイムによる共通問題の設計」という特集です。ここでプログラミング・パラダイムとして取り上げられているのは，1)オブジェクト指向，2)論理型，3)属性文法，4)モジュラプログラミング，5)ストリーム型，6)関数型，の6つです。この特集のユニークなのは，在庫管理というおなじテーマをいろいろなプログラミング・パラダイムで解いていることです。

しかし，この特集の冒頭で，プログラミング・パラダイム＝プログラミング技法，としていることがまったく残念です。パラダイムということばがここでは単なるテクニックを表すものとして扱われているような気がするのです。クーンの論じたパラダイムということばは，20年たってずいぶんな扱いを受けているといえるでしょう。

なぜこれほどのギャップが生ずるのかを考える場合に，まず押さえなければならないのは，天動説か地動説かという選択とは違って，種々の言語のひとつだけが正しいわけではないことです。そもそも，現在のプログラミング言語が扱っている対象が，まだまだ限られているということにも大きな原因があるのかもしれません。知的でお

茶目な計算機への進化がもっともっと進めば，人がものごとを認識するとき必要となる枠組みのようなものも処理できるマシンが生まれてくることになると思います。

さて，最近の研究をほんのちょっとだけ見てみることにしましょう。10月中旬に九州工業大学で開かれた情報処理学会の全国大会で発表された論文集をざっと見て，論文のタイトルに含まれることばのうち，ちょっとでも「パラダイムっぽい」匂いがするものを拾ってみました。

「超論理推論，マルチメディア，フレキシブルハイパーメディアシステム，場と一体化したプロセスの概念，最小例外世界に着目した条件論理，信念に基づく非単調知識処理……」

うーむ，パラダイムに近いものを選んでいるのか，僕にとって意味不明なものを選んでいるのか，意味不明になってきたのでやめましょう。

パラダイムシフトは可能か?

計算機アーキテクチャに関して，パラダイムシフト的なことはどんなことがあったでしょうか？ データフローマシン，仮想メモリ，RISCなどがそうなのでしょうか？

まあそれほど大はずれということはないかもしれません。ここでもう少し最近の例として，トレーススケジューリングに基づくVLIW(Very Long Instruction Word)アーキテクチャでも挙げてみることにしましょう。

最近，大学レベルだけでなく，多くのメーカーも取り組み始めたこのアーキテクチャは，文字どおり，1つの命令のビット幅が数百から1000以上という驚きに値するものです。そして，異常に長いビット幅をもつ命令ひとつで，つまり並列に制御してしまうというものです。

単にこれだけならば，とっくの昔に考えついてもよさそうですが，このアーキテクチャを支える「トレース・スケジューリング」という手法が極めて革新的であったからこそ，いまVLIWが隆盛を極めようとしているのです。「トレース・スケジューリング」とは，並列化にとってガンであった，条件分岐の壁を取っ払ってしまおうという手法です。このVLIWについては近いうちにもう少し詳しく紹介したいと思います。

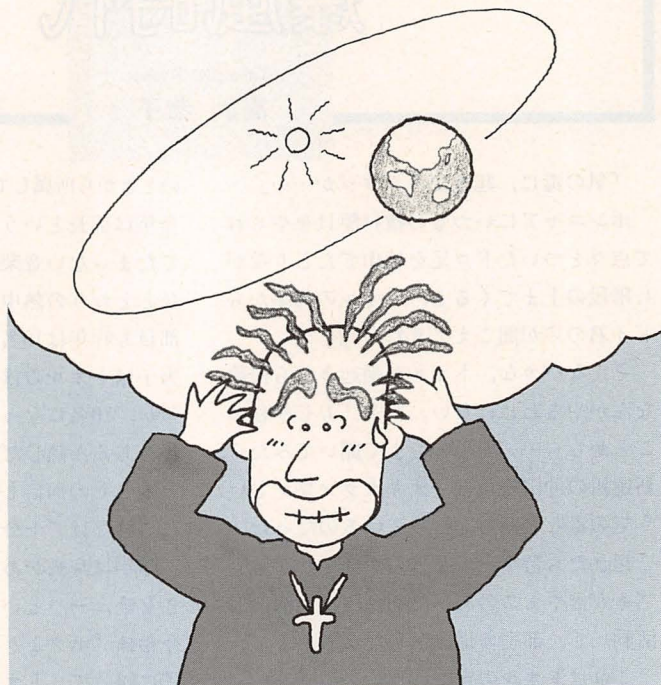
いずれにせよ，天動説から地動説へのシフトというようなレベルの話はどうもまだ生まれていないし，またこれからもすぐには生まれにくそうに思われます。フォン・ノイマンアーキテクチャがそれほどすばらしいということなのでしょうか？ あるいは，単に計算機が生まれてまだほんの少ししかたっていないからということなのでしょうか？ 僕としては，後者であると信じて研究を続けているわけです。

本当のパラダイム

僕自身に関しては，「パラダイムっぽい」ことをやっていたのは，やはり卒論のときでしょう。前に紹介したのでここではあまり触れませんが，簡単にいえば，大昔ことばというものがなかったような原初において，どのように言語が成立したかということです。

その際のキーワードが「自己組織化」です。このようにすれば相手と通信し(話し)意味を理解できるのだということを外から教えたりするのではなく，何らかの本能的なもののみ最小限用意し，基本的には，すべて自分自身でことばを作り上げていくということです。

「お茶目な計算機」を目指す際に特に重要となるのがこの自己組織化であるといいきっていいでしょう。この自己組織化というパラダイムは，いまニューラルネットでたいへんなブームを引き起こしているということはあらためていうことでもないでしょう。

その後，博士過程から現在にかけて細々とやり続けているものをひとつ紹介しておきましょう。「自己組織化」ほど派手ではありませんし，まだ広めていないのですが，「即時実行」[4]ということです。要するに，「あることをスピーディに行うのに，必要な準備などをグダグダするくらいならば，準備不足でもいいからすぐに取りかかりなさい！」ということです。これには，ソフトウェア側から見た，並列ハードウェアの研究に対する疑念が，案外こもっていたりするのです。

僕としては，なるべく明確で普遍的な概念の枠たるパラダイムを計算機の世界にも求めたいと思っています。なぜならば，単に計算機の箱の中の話，プログラミング技法の話にとどまらず，より普遍的なもののほうが，結果的にはすばらしいものだったということになるのではないかと，盲信的に信じているからです。

参考文献

1) 村上陽一郎：クーン，現代思想の109人(現代思想臨時増刊)，pp.230-231，青土社（1978）。
2) C＋Fコミュニケーションズ編著：パラダイム・ブック，日本実業出版社（1986）。
3) 特集：新しいプログラミング・パラダイムによる共通問題の設計，情報処理，Vol.26，No.5，pp.458-520（1985）。
4) 有田隆也ほか：高水準言語プログラムの即時実行方式について，電子情報通信学会技術研究報告，CPSY89-13（1989）。

第42回

猫とコンピュータ

爆風時代

Takazawa Kyoko
高沢 恭子

秋といえば1年間のうちで学校行事がもっともさかんに行われる頃。秋の遠足に運動会，文化祭や学芸発表会などがめじろ押しです。キョウコさんの家でも，トオルくんがめまぐるしい毎日を送っているようです。

「気の毒に，起きたらソウジか……」

ホンニャアにいつもの抜け駆けをやられて点々とついたドロ足を雑巾でたどりながら階段の上までくると，トオルの部屋からドイ君の声が聞こえてきた。

そんなバカな，トオルが朝起きたら掃除なんかやるわけはない。ン？ もしや私のことかしら……と思ってよく聞いてみたら，新選組の沖田総司は，オキタラソウジという気の毒な名前だと言っているのだった。

「起きたら習字と，どっちがいいかなぁ」

「そう言やぁこのあいだ朝練（授業前の部活動）で，部室の掃除をしたなぁ」

これはトオルの声だ。

おと(音)もだち

中学校の3年間の歳月は，なんというかけがえのない日々であることか。花の種を何万と背負って，いたるところに蒔きながら，水と光を求めて過ごしているかのようだ。毎日が新しく，毎日が試みの連続で，決して後戻りがない。パソコンの歩みもかないはしない。それにしては短い，早い3年間であると思わせられる。もうトオルも2年生だ。中学2年生のパワー満開の活動ぶりには，まったく目をみはる。中学校生活においての中心の役目を果たすのが2年生なのだと，先輩から学び，先生方からも望まれるから自覚もある。

「生徒会の役員は今期だけで辞めることにしたからね」とトオルが先日言った。よい経験になるからやってみるよう先生方からのおすすめがあって立候補したのだが，「生徒会の仕事をしていると，いかに部活がおろそかになるかわかったんだ」とさらに言った。何かを始めたら納得ができるところまで続けてみるのが，いままでのトオルのやり方だったから，よほど心に感じることがあったのだ。部活というのは1年生のときから所属しているブラスバンド部だが，今年は部長ということで，ただでさえ好きでたまらない音楽に責任感が上乗せされて，ひととおりの熱中ぶりではなくなっている。部員も昨年は10名たらずだったのが40名，男子はトオルのほかはサトウ君だけだったのが，10名になった。ふだんの練習もみんなもちろん熱心だがセレモニーや発表会となるとその何倍もの練習に励んで，たしかにこれだけで十分に多忙なのだ。

昨年は身長があるというので，とても大きなチューバという管楽器を吹いていたが，今年は「ボクより上手」というほかの男の子に譲って，トオルはドラムスの担当になった。ドラムは先生も専門外で苦手というのだが，音楽好きのトオルにとってコワい楽器はないのかもしれない。新宿の家の2階の一室には，私の弟が学生時代に叩いていたドラムセットが置いてあって，小さなときからイタズラしていた親しみもあるらしく，例によって参考書をあれこれ買い込み，徹底的にのめりこみをつづけている。

音楽も演奏を楽しもうとしたらやはりひとりより複数が数倍楽しそうだ。自分の受け持った楽器は初めから明らかに全体の中の部分であり，そのパートを正しく首尾よくプログラムどおり実行することで，ひとつの曲を作り上げ表現する。1人ひとりが体内に持ったリズムをひとつにして，同一の時間内を緊張と相互信頼で過ごすなんて，音楽の仲間だけにしかわからない，強烈な連帯の喜びなのかもしれない。

ブラスバンドはいかにも若さを全開放したという感じのパレード，式典向けの楽隊だが，親しみやすい楽器は学校用の音楽としてもふさわしく，楽しい中にチームワークも満喫できる理想の一面を持っていると思う。トオルでなくても熱中して不思議はなさそうだ。

イベントマニア?

それまで並行してなんとか過ごしていた生徒会役員の仕事が，秋になると「学芸発表会」という一大行事をめざしてたいへんに忙しくなってきた。これは毎年，全校がクラス単位で演劇や演奏，ビデオ，展示物などを発表し合うもので，実行委員会が設置され，生徒会とのジョイントで開催されるのだ。3学年の各クラスに一部のクラブも加わると参加数は28くらいになり，各担任の先生も創作の一員となって，これは例年たいへんなイベントなのだ。

生徒会役員と実行委員は開催準備の打ち合わせで連日時間をかけるようになってきた。そのうえ，クラスのメンバーとしては演目選びに始まる，上演のための協力をしなければならない。

昨年トオルのいたクラスは，いじめの問題を扱った創作ドラマで大好評を得たが，こんどはなかなか演目が決まらない。担任の先生はトオルが小さいころ書いた猫の冒険小説をミュージカルにしようと提案されたりもしたのだが，結局落ち着いたのは英語劇『PEACH BOY』（桃太郎）だった。

発表会にはブラスバンド部も参加することになっているので，そちらでは2つの新しい曲「ライディーン」と「JUST ONE VICTORY」を練習しなければならない。

「生徒会とブラバンで時間がとれないから，クラスの劇はかんべんして……」と言ったはずなのに，キャスティングの候補を班ごとに提出したら，おばあさん以外の役に全部トオルの名前があった。いったいどんなふうにクラスメートから思われているのか。とうとうおじいさんの役をやることになった。桃太郎はゴムまりみたいにかわいい女の子だ。

2日間の発表会で，彼は運営側として進

行と紹介の役も半分ほど務めるというし，こんなムチャクチャな話はスケジュールだけでも成り立つはずはない。いつまでも暑さの残る秋だったが，想像どおりホコリと汗の毎日が続いた。徒歩50秒の目の前にある中学校から１日に何度も帰宅しては，必要な楽譜や書類やオーディオの部品などを持ってまた出かけていく。ほんとに帰宅するのは７時に近かった。カバンの中もおびただしい数の手書きの楽譜と，桃太郎の台本と生徒会のプリント類がスクランブルで，教科書は威厳がない。聞けばこのさなかに，３年生の３つのクラスから「DIAMONDS」，「リゾ・ラバ」，「GLORIA」の編曲をそれぞれ頼まれているのだという。

「英語劇のセリフ，覚えられるかなぁ」と言いながらあまり心配そうでもないし，まあ，あんなに生き生きしているのだからやらせておこうと見守って過ごした。

学芸発表会は父母も見学して無事終了したが，トオルはいろいろ考えた。最大の後悔はブラスバンド部の練習を何回となく削らなければならなかったことだった。そしてそれはもっと大きな学校全体の活動をひきいていく生徒会のためにやむを得なかったことで，自分はそちらも十分重視していたこと。ふたつの責任を承知していながら果たせなかった無責任は，もう繰り返したくないと思った。

「それでね，ドイ君がこんど立候補するって……」

「ああ，そう，後期からはふたりで協力したいって言ってたのにね」

「ドイ君ならだいじょうぶだよ，副会長に立候補するって」

生徒会も３年生からバトンタッチされる季節がきたのだ。よく考え，自分で意義を見つけたことを大切にしてがんばってくれたらそれでいいと思う。

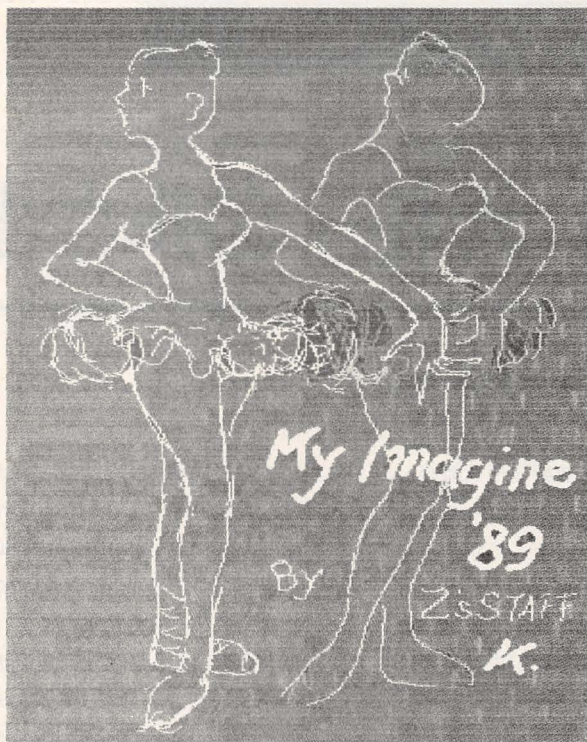
実験CG

池田満寿夫さんが初めてパソコンで描いた絵という言葉に誘われて，こぬか雨の降る10月なかばの日曜日，新宿駅東口のコニカプラザ（ギャラリー）に夫とふたり散歩がてら出かけた。「自然・環境・人・デジタルアート」をテーマに，日米のコンピュータグラフィックを160点集めた「IMAGINE TOKYO '89」展である。トオルは体育の日に開かれる区民まつりのパレードで，30あまりの中学校がブラスバンドの大合奏をするというので，練習のための日曜登校だった。

コニカプラザのお隣はいまや名高い「スタジオアルタ」，この日も入り口付近はヤングでいっぱいだったが，こちらはもったいないほどの静けさだ。もったいないと思ったのはなかなか新しい試みをほどこした作品が，ほとんど無人に近い状態で鑑賞できたからだ。作者もプロの画家，写真家，デザイナーたちで，コンピュータグラフィックといっても，純粋にパソコンだけで描いたのものばかりでなく，パソコンの機能をひとつの描法またテクニックとして自分の作品に取り入れ，表現をひろげようとしているものがたくさんあった。

たとえば写真家がある情景を素材として選び，パソコンのプログラマが撮影の段階から立ち会って創作イメージの了解をしあう。できあがった写真の作品にプログラマがコンピュータグラフィックのテクニックを融合させて，イメージの完成に近づける。また，CGのタイルパターンふうの描法で人体を描き，画面写真を拡大して印画紙で大画面をつくり，その上にコラージュをほどこしたもの，あるいはやはりCGの拡大画面に絵の具でペインティングを加えて作品としたものなど。

この中で，マウスを扱ったのは初めてという池田満寿夫さん以下日本の８人の芸術家の作品が，「パソコン版画」と名づけられて出品された。パソコンで描いた原画がプリンタで何枚も再生されるのを版画に見たててのネーミングだそうだ。華道家の勅使河原宏さんも参加されていたが，大胆で端的な画面が作風を象徴していて，CGばなれした魅力があった。

この催しについてはパンフレットが作られておらず，作品ごとにネームプレートを兼ねたかんたんなコメントがあるだけ，フロアにいた係の女性たちも展示内容については詳しくないので，とても物足りなくて心残りだった。ただし，パソコンやCG，自分のマシンについてのアンケートが用意されていて，これにふたりでそれぞれ回答したら，おみやげにマクセルのフロッピーを１枚ずつちょうだいした。池田氏のMacIIで描いた作品のポスターをせめてもの記念に買い求めたけれど，入場無料なんだし外にいる若い人たちも覗いてみたらいいのにな，なんて思いながら帰ってきたものだ。

陽のあたるパソコン

パソコンというものが，ようやく明るいところに引き出されてきたかなと実感するこのごろで，ふつうの人たちの関心もじわじわ高まってきた。

パソコンに関する記事や情報も以前とはくらべものにならないほど多くなったし，これから始めようとする人たちのための親切な雑誌もできてきた。JRの駅にはパソコン教室を設けているところがいくつもあるそうだ。

「互換性」の問題なども深刻に言われながら，やはりパソコンはずいぶん使いやすくなったのかなと思っている折，あるメーカーの企画する催しで，女性の方たちを対象に「楽しめるパソコン」風のテーマで話をしてほしいという依頼がきた。時は11月３日，会場は池袋サンシャインビルの一室。さあ，たいへん。

Oh!X 2周年特別企画

X68000にガイガーカウンタをつなぐ

素粒子の声が聞こえる

恐ろしい放射線，といってもそれも素粒子の一形態にすぎません。ここでは原子の世界からの声を聞くためガイガーカウンタをX68000につないでみましょう。星の綺麗な夜，原因不明で再現性のない暴走を起こしたとき，プログラマはそっとつぶやきます。「あれ，バスに宇宙線が命中したかな」

Kuwano Masahiko
桒野 雅彦

放射能や放射線という言葉はごく日常的な言葉になっていますが，原爆や原発のせいもあってか，そのイメージはかなり否定的なものになってしまっているようです。目に見えない恐ろしいものとしてのイメージだけが先行して，放射線そのものが悪玉のようにとられてしまうのは残念な気もします。

放射性同位元素が地層などの年代測定や医療目的に使われていることはよく知られた事実ですし，それ以上に貴重なのは放射線が原子から聞こえてくる，ささやくようなメッセージであるということです。かすかに聞こえるその声を（少しずつではありますが），聞くことができるようになると私たちがそれまで抱いていた「物」に対するイメージを根本から揺さぶるような結果が次々と知られることになりました。

今回は，この小さな宇宙からのメッセージをコンピュータで聞いてみることにしましょう。

放射線

放射線，放射能といった言葉はほとんど一般用語にもなっていますが，一応定義らしきことをいっておきましょう。一般にいわれる放射線とはほかの原子や分子にぶつかって，それをイオン化させることのできるものを指しています。ですから，赤外線や可視光線などは放射線としては扱われません。また，放射能というのは放射線を出す能力のことを指します。

言葉は生き物（ナマモノ）ですから，放射線と同義語としても使われることも少なくありませんが，熟語になるときはわりとこの定義らしきものに準拠する場合が多いようです。たとえば，放射線汚染ではなく放射能汚染ですし，放射線源であって，放射能源とはいいません。頼みもしないのに余計な「放射線を出す能力」がつけられてしまったから放射能汚染であり，「放射線の源」だから放射線源と読んでみると，納得できるものがあります。

さて，放射線をさらに分類すると，電荷を帯びた高速の粒子と，中性子のように電荷を帯びていない粒子や波長の非常に短い電磁波（つまり，電波）に分けられます。

荷電粒子は直接原子や分子をイオン化する能力があります。このような放射線を直接電離放射線といいます。一方，電気的に中性である中性子や電磁波などはそれ自身では電離能力はなく，ほかの物質とのあいだでなんらかの相互作用をした結果として発生した荷電粒子が電離作用を持つので，間接電離放射線といいます。

荷電粒子のほうでよく知られているのはα線とβ線でしょう。このほかにも，陽子線，核分裂片などがあります。荷電粒子とは，電子や原子核が裸で飛んでいるものと考えておいてもよいでしょう。

一方，放射線に含められている電磁波としてはγ線，X線などがあります。この両者は波長が違うだけのことですが，電磁波というのは波長によってその性質が大きく変わっていくことや歴史的な経緯などもあって，別々の名前になっています。

そのようなわけでX線とγ線の境界はあまりはっきりしていません。あえて言葉にすれば，波長がすごく短いのがX線で，波長がものすごく短いのがγ線というよりないようです。歴史的に見るとX線の領域が広がり，γ線の領域に進出していく傾向にあるようです。

核分裂と核融合

ここで，ちょっと寄り道をして，核分裂と核融合についても触れておきましょう。世の中で使われている原子炉では，ウラン235やプルトニウム239などの大きな原子核が分裂するときに放出されるエネルギーを使っているわけです。一方，ちょっと前に常温核融合，つまり原子核同士がくっついて新しい核種になる反応が常温で起きたとかいうことでずいぶん騒がれました。核融合は複数の原子核がくっついてひとつの重い原子核になる反応で，いまのところ重水素やトリチウムといった通常の水素原子の原子核（要するに陽子です）に中性子を加えた重たい水素原子（重水素）が2つくっついてヘリウムになるという反応を起こしやすいのではないかと見られています。核融合炉というのは，このときに放出されるエネルギーを利用しようという魂胆です。うまくいけば，海水の中に含まれるほとんど無尽蔵といってよい量の重水が使えます

放射性同位体（同位元素）

原子核が励起され，放射能を持ってから，放射線を出しつつ別の核種になっていく過程には相当時間がかかる場合が少なくありません。このようなものでは，延々と放射線を出し続けることになるわけです。このようなものを放射性であるといいます。

同じ原子番号でも質量数が違う原子のうち，放射性である原子を放射性同位体といいます。放射線を発生することで，その原子はより安定な原子に変化（崩壊）していきます。崩壊がいつ起こるかはまったく不確定で，確率でしか求めることはできません。このときの確率は一定量の放射性物質が平均して半分の量に崩壊するまでの期間（半減期）に変えて表現されます。

ので，夢のエネルギー源として注目されているわけです。

ここで，ちょっと妙に感じるのは，核「分裂」でも核「融合」でもエネルギーが取り出されるということです。どちらでもエネルギーが取り出せるというのは，かなり不思議な気がするでしょう。もし，本当にくっつけても離れてもエネルギーが出るというのであれば，それこそエネルギー保存則に対する大胆な挑戦です。そんなことができたら永久機関になってしまいます（特許をとれば大儲け……です。もっとも，日本の特許庁では永久機関といって出しただけで即，ボツにされるそうです）。

これを解くカギは，原子の質量にありました。ここに，ちょっとした資料があります。単位がamuとなっていますが，これは原子質量単位というもので，1 amuが1.66×10^{-27} kgにあたります。

陽子1個の静止質量　1.007276amu
中性子1個の静止質量　1.008665amu
電子1個の静止質量　0.000549amu

さらに，

重水素原子の静止質量　2.014102amu
炭素原子の静止質量　12.000000amu

さて，ちょっと計算してみましょう（X68000を使っている人はOPT.1＋OPT.2で電卓が使えますね）。重水素の原子は，陽子，中性子，電子がそれぞれ1個ずつでできています。したがって，これらの和は1.007276＋1.008665＋0.000549＝2.016490 amuとなります。ところが，実測してみると重水素の原子の質量は2.014102amuです。その差0.002388amu，0.1％ほど質量が小さくなってしまっているのです。さらに炭素のほうでも調べてみましょう。炭素は陽子，

わかりやすい放射線紳士録

○α線

α線は陽子が2つと中性子が2つくっついたもので，要するにヘリウムの原子核が電子を引き連れないまま飛んでいるものです。図1を見てもわかるように，質量の割にヘリウムの結合エネルギーが飛び抜けて高いことがわかります。核分裂のときに核子がα線として放出されやすいのはこのかたちが安定であるためです。

α線の持っているエネルギーはだいたい数MeV（1エレクトロンボルトは真空中で電子が1Vの電位差によって受け取る運動エネルギー，$1eV=1.60\times10^{-13}$J）程度であり，重いうえに電荷が大きく（陽子を2つ持っている），電離能力も優れています。

まわりに影響を与えやすい分，物質の透過能力は少なく，大気中でも1mmいくかどうかという程度です。散乱されることもあまりありません。つまり，紙1枚分くらいの大気を通過する間にそのエネルギーのほとんどを失ってしまいます。「α線は紙1枚で阻止できる」とよくいわれるのは紙の繊維の目を通過できないということではなく，紙1枚の大気を透過することができないという意味だったのでした。

○β線

α線は原子核でしたが，β線のほうは電子です。普通，電子はマイナスの電荷を持っていますが，β線として飛んでいるときにはプラスの電荷を持っているものも，ちょくちょく見つかります。マイナスの電荷を持ったほうをβ^-，プラスのほうをβ^+と呼んで区別するときもありますが，黙ってβ線というときはβ^-を指します。

β^+線は反粒子，いわば反物質みたいなものですから，原子中の普通の電子とぶつかったりすると相手もろとも消滅してしまいます。世の中の原子は，中心のごく一部が原子核で，あとは全部電子みたいなものですから，β^+線がいつまでも無事でいられることはまずありえません。もちろん，「消える」とはいっても，エネルギー保存則は成り立っているので，「跡形もなく」消えるようなことはしません。ぶつかった方向の両方に約0.51MeVの電磁波（γ線）が放出されます。このエネルギー値は電子の静止エネルギー，有名な$E=mc^2$の式によって求まるエネルギー分が放出されるわけです。

β線はいろいろな過程で飛び出してきますので，持つエネルギーもかなりの幅を持っていて，数keVから数十MeV程度までばらつきます。また，持っている電荷の割に軽く小さいもので，電離能力はそれほど大きくはありません。物質透過能力に関してはα線よりもだいぶ優れていますが，それでもせいぜいアルミ板で数mm，鉛では1mm程度です。

○X線

X線は，原子の内殻電子がエネルギー準位の高い外側の軌道から低い内側の軌道に移るときに，そのエネルギーの差分を電磁波として放出したものと，荷電粒子が電磁場などで急減速を受けたときに放出する（制動放射）ものがあります。

量子論によると，振動数νの電磁波のエネルギーEは$E=h\nu$（hはプランク定数：$6.6256E^{-23}$ erg・s）$=hc/\lambda$（cは光速度(m/s)，λは波長（m））で与えられますから，振動数が大きい，すなわち波長が短いほど大きなエネルギーを持つことになります。内殻電子の軌道変更が少なく，それに伴うエネルギーの吸収，放出では比較的エネルギー変化が少ないため，可視光線や紫外線になります。X線くらいの波長になるとエネルギーも結構な大きさになりますから，軌道をひとつや2つ移動したくらいでは話が合いません。つまり，X線が放出されたということはいくつもの軌道を一気に飛び越えて移動したということを意味します。

ところで，電子の軌道というのは飛び飛びの場所にしかないため，この原理で発生するX線のエネルギーというのも，当然のことながらそれに応じた不連続な値しか取れません。つまり，内殻電子の軌道変化によって発生するX線の波長は飛び飛びの値，線スペクトルを取ることになります。このような原理で発生するX線を固有X線（特性X線）といいます。

一方，荷電粒子の急減速によるほうはこのような不連続な値になる理由はありません。スペクトルは当然，連続スペクトルになります。こちらを連続X線と呼びます。

○γ線

γ線は，X線よりもさらに波長が短い電磁波です。持っているエネルギーはGeV（1G＝1,000,000k）程度と，とてつもなく大きくなります。これくらいになると，内殻電子の軌道変更くらいでは説明できません。

γ線は，原子核内部での変化や先ほど触れた電子の消滅など，派手なイベントが起こったときに発生します。

X線がいわゆるレントゲン撮影などで使われることからもわかるように，X線やγ線は透過能力が大きく，電離作用はかなり小さいものです。X線，γ線は直接電離能力がなく，持っているエネルギーをいったん物質の中の電子に引き渡し，その電子が電離作用をします。この受け渡しは，光電効果，電子対の生成，コンプトン効果などによっているのですが，これらがうまくいく確率というのがそれほど高くないため，電離作用がさほど大きくならないのです。

○核分裂片

もとの原子核が不安定であったために，複数の核に分裂してしまったものです。このとき，より安定な状態に移行することになるために，その差分のエネルギーが放出されます。ウラン235やプルトニウム239などの原子核が分裂するときのエネルギーを一気に取り出したのが原爆，濃縮率を下げてゆっくりと取り出すようにしたのが原発です。

○中性子線

中性子は，陽子とほぼ同じくらいの重さを持っていながら電気的には中性です。核分裂が起こるときは必ず放出されます。中性子は，陽子と共に核を構成しているときは安定なのですが，単独で存在するときわめて不安定でβ線（電子）を放出して陽子に変身してしまいます。

○陽子線

α線がヘリウムの原子核とするなら，こちらは水素の原子核となります。中性子線が，水素原子のために散乱され，反跳をうけたときに放出されることがあります。

中性子，電子がともに6個ずつでできています。同じように計算すると12.09894amuとなりますが，実際には12.000000amuなのです。今度は0.8％もの質量が行方不明になってしまいました。

科学者としては行方不明です，ではすまされないわけで，やはり合理的な考え方を打ち立てなくてはなりません。

ここで「核の結合エネルギー」という概念が導入されるわけです。質量とエネルギーの関係は，例によって相対性理論でお馴染み$E=mc^2$の例の式ですね。減ってしまった質量はエネルギーとして放出されてしまったと見るわけです。つまり，陽子と中性子がバラバラになっているときよりも，互いにくっついて原子核を構成しているときのほうがエネルギー的に低い状態であり，その差分が結合エネルギーであるとするのです。これは，位置エネルギーにも似ていて納得しやすい概念でしょう。互いに結合するときにはこの分のエネルギーが，外部に放出されます。もちろん，くっついてしまった陽子と中性子を再びばらばらにするためには当然，結合エネルギーに相当するエネルギーを与えなくてはなりません。

図1

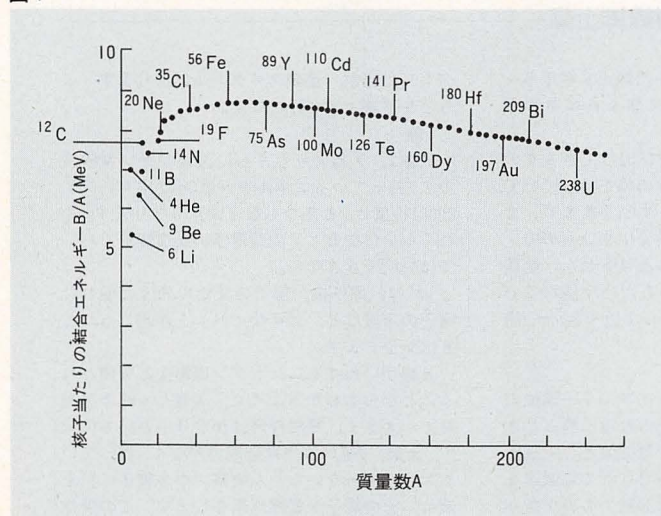
まとめると，ばらばらの陽子と中性子が結合するときには，結合エネルギー分のエネルギーを放出し，その分軽くなるというわけです。

これによれば，どんなものであっても，その原子の構成と正確な質量さえわかれば結合エネルギーに相当する分がわかることになります。このノリで，いろいろな原子について結合エネルギーを調べ，それを陽子と中性子の数の合計で割り算すると，陽子，中性子1個当たりの結合エネルギーがわかります。横軸に質量数，縦軸に1個当たりの結合エネルギーをプロットしていくとどうなるでしょうか（図1参照）。

もし，核子（陽子，または中性子）同士の結合力が広範囲に渡ってきくのであれば，新たに加えられた核子はほかのすべての核子と結合するわけですから，グラフは右上がり，すなわち原子番号が増すにつれて核子1個当たりの結合エネルギーも増えていくはずです。ところが，実際には鉄（Fe：原子番号56）のあたりにピークがあってあとはなだらかに減少する傾向にあります。核子同士の相互作用が働くのは割と狭い範囲であることがわかります。

ごく狭い範囲同士でしか結合できないのですから，原子核がどんどん大きくなっていくと，全体がひとつにまとまっていることも難しくなり，ずいぶんと不安定な状態になります。ウラン235の原子核などはあまりに巨大になりすぎたために，もはや綺麗な球形でおとなしくしていられず，不格好な形でうごめいているようです。こうなると，きっかけ（ほんの少しの励起）さえあれば分裂してより安定になろうとするのは理の当然でしょう。

これで，核融合と核分裂の謎解きができたようです。核子1個当たりの結合エネルギーが上がる方向に向かっていくときにエネルギーが出るわけです。核子1個当たりの結合エネルギーがいちばん大きいところ，平たくいってしまえば核子がいちばんよくまとまったところが鉄であり，ほかの元素から鉄の方向に向かうとき（もちろん，多少のでこぼこはありますので，必ず鉄の方向というわけでもないですが）には結合エネルギーの差分が放出されるというわけです。

そのため，重水素やトリチウムがヘリウムになる反応，ウランやプルトニウムが核分裂を起こしてより軽い原子になる反応などはエネルギーの放出を伴うわけです。

◎メモ1：電磁波

電磁波のうち，波長の長いものは電波と呼ばれる領域になります。波長がうんと長い領域は長波（LF），ラジオに使われているような周波数3000kHz（3MHz：波長100m以上）以下のあたりを中波（MF），周波数が3MHzから30MHz（波長100〜10m）は短波（HF）と呼びます。ひところはやったBCLは海外短波放送がターゲットでした。短波は上空にある電離層によって反射されるため，地球の裏側からでも届くのです。これについては，ちょっといい話があります。

電離層が見つかるまでは，短波はごく近くの通信にしか使えないと思われており，長波，中波が通信の主体でした。そんな中でアマチュアの無線家が電離層反射による伝播があることを見つけ，これが情報伝達の大きな革命になりました。現在では電波が多くの目的で使用されるようになり，決して余裕があるとはいえない状態ですが，それでもアマチュア無線専用の周波数が，いろいろな周波数帯（現在，主に使われているだけでも10バンド以上）で割り当てられているのは，この功績がなによりも大きかったといわれています。

さて，寄り道はこれくらいにしてさらに周波数の短い領域を見ていきましょう。テレビなどにも使われる30MHzから300MHz（波長10〜1m）のあたりを超短波（VHF），さらに3GHz（波長10cm）くらいまでをUHF，30GHz（波長1cm）あたりまでがSHF……となっていきます。UHF領域はローカルな放送局で，またSHF領域は衛星放送でも利用されています。SHFくらいになると伝播のしかたも光に似てきますし，雨などの影響も受けやすくなってきますので，実務的に使えるのはこのあたりまででしょう。

もっと波長を短くしていくと，ついには熱として感じられる領域である赤外線となります。さらに光として目に見える可視光線になり，化学作用の強い紫外線として日焼け，シミ，ソバカスの原因ともなり……と，波長が変わるにつれて次々と性格が変わってきます。X線は紫外線よりも波長の短い領域，10^{-8} m程度以下のところから10^{-10} m程度のところ（ちょうど原子ひとつ分くらいの長さ）になります。さらにずっと波長の短い領域はすべてγ線と呼ぶようになっています。

放射線計測

放射線は，たとえばβ線であれば電子1個，α線ならヘリウムの原子核1個というぐあいに原子を構成する単位のようなものであるために直接の計測はほとんど不可能で，あくまでも間接的に確認することしかできません。

放射線の計測はいろいろな方法が考え出され，使われていますが，いずれも直接放射線そのものを計測するものではなく，放射線がほかの物質に当たった結果として起こる現象を測定しています。

それでは，代表的な放射線計測の方法を見てみることにしましょう。

○霧箱

一面をガラス張りにして，中が見えるようにしたシリンダで，ピストンを急に引っ張る（断熱膨張する）と温度が下がり，中の水蒸気が過飽和状態になります。ここに荷電粒子が飛び込むと，その通り道にそってイオン対ができます。そこに水蒸気が凝縮するため，外から白い霧の線が見えることになります。簡単な構造ですが1910年頃には主要な実験装置でした。

放射線源からピシピシと放射状に線が飛ぶ様子がリアルタイムに観測できるのは大きなメリットです。運がよければ，飛び込んだ荷電粒子（主にα線）が霧箱の中のほかの原子（窒素など）とぶつかって，陽子が飛び出したりするのを見ることもできます。

かなり簡単な構造であり，物理的に壊れにくいので学校の設備として持っているところも多いでしょう。いまなら，断熱膨張など使わなくてもドライアイスなどという便利なものがあるので，箱だけ作って底に黒い布などを引いてアルコールを振り掛け，電極を入れたものをドライアイスで冷却して（冷やしすぎないように）電極に数百V程度をかけてよけいなイオンをとっぱらえば，一丁あがりです。

○シンチレーション計数管

シンチレーション（蛍光）計数管というのは，荷電粒子によって励起されたあと，元に戻る際に光を出すような結晶（NaIやCsIなど）などを使い，その光を捕らえようというものです。ただ，この光はかなり弱いものなので直接センサに捕まえさせるのはちょっと酷……ということで市販品では光電子倍増管とペアにして円筒型の容器に入れているのが普通です。

シンチレータとしては無機結晶，有機結晶，液体などいろいろなものがあり，目的別に使い分けられています。

○電離箱，比例計数管，ガイガー・ミューラー計数管

これら3種類の検出器はいずれも放射線による電離作用を使うものです。いずれも薄い不活性ガスなどを封入した円筒型の金属の筒の中心に電極を置いてあり，円筒との間に電圧をかけてあります。電離放射線が飛び込んで，中のガスが電子とイオンに電離されます。ここで電極間に電圧がかけられているとどうなるでしょう。

電圧が低いと，せっかく作られた電子とイオンはその場で再び結合してしまい，あまり面白いことは起こりません。ここで少しずつ電圧を上げていくと，電離された電子とイオンがそれぞれ電極（電子は陽極，イオンは陰極）に移動します。移動してくれば……そう，電流が流れるわけです。これが電離箱です。

◎メモ2:光電効果などについて

光電効果は，金属に光を当てると，表面から電子（光電子と呼ばれます）が飛び出してくる現象です。波長が一定以下でないと電子は放出されず，また光の強度を上げても飛び出してくる電子の数が増えるだけで，電子1つひとつのエネルギーは変化しません。かのアインシュタインがノーベル賞を受賞したのは光電効果の研究によるものでした。相対性理論のほうは受賞対象にならなかったという話も残っています。

電子対生成は，高エネルギーのγ線によって，電子と陽電子ができる現象です。陽電子のほうはβ^+のときと同じように，すぐにほかの電子とぶつかり，γ線を放出して消滅してしまいます。

コンプトン効果は，X線が物質に当たって散乱されたとき，散乱されたX線の波長が入射したX線の波長よりも長くなる現象です。この差分が当たった相手に受け渡されたことになります。

光電効果やコンプトン効果も，一見したところどうということのないような現象ですが，これを説明するには光を単なる波動とするような古典理論ではだめで，$h\nu$のエネルギーを持った光量子と考える量子理論が必要になります。

放射線関連の単位について

放射線関連の単位というのは実に種類が多く，混乱しやすいのでここでまとめてみました。単位をその種別ごとに分類すると次のようになります。

単位時間当たりの原子崩壊数に注目したもの

　キュリー（Ci），ベクレル（Bq）

X線やγ線によって空気中に作られるイオンの総数に注目したもの（照射線量）

　レントゲン（R）

放射線で照射された物質の単位質量当たりの吸収エネルギー（吸収線量）に注目したもの

　ラド（rad），グレイ（Gy）

吸収線量に生態に及ぼす効果を考慮した係数を掛けたもの（線量当量）

　レム（rem），シーベルト（Sy）

2つ以上並んでいるものは，左側が伝統的な単位系，右が国際単位系（SI）による単位系です。レントゲン（R）についてはSIでは特に単位を作らず，単にC/kg（クーロン/キログラム）としているだけです。

キュリー（Ci）は，いわずとしれた，キュリー夫妻の発見したラジウム約1gの崩壊数で，1Ci＝37000000000崩壊/秒です。1Ciというのはかなり大きい値であり，現実的にはmCi（ミリキュリー）やμCi（マイクロキュリー）として使われます。SIでは1崩壊/秒を1Bqとして定めました。

レントゲン（R）は乾燥した空気1kg当たり2.58×10^{-4}クーロンの電荷を作る（1.6×10^{15}個のイオンを作る）分のX線，γ線の線量のことです。普通は1時間当たりのレントゲン数などが単位として使われます。

ラド（rad）は，1グラム当たり100erg（10^{-5}J）のエネルギーを吸収したとき，1radとしています。SIでは1kg当たり1Jの吸収，つまり10^2ラドを1Gyとしました。

レムは，ラドに係数（線質係数Q）を掛けたものです。Qの値はX線，γ線，β線では1，エネルギー不明のα線や電荷不明の粒子では20，電荷のわかっているα線では10としています。SIではグレイにQを掛けたものをSyとしています。

放射線関連は，被爆による生体や環境への影響ということが大きくクローズアップされているためか，単位もほかの物理現象とは違って随分といろいろな単位があります。特にここ数年，SI単位系への切り替えが進むにつれ，素人にはいよいよわけがわからなくなりそうです。

さらに電圧が上がると，最初に生まれた電子（1次電子）が陽極に移動するまでのあいだにほかの気体の分子，原子を電離することができるほどにまで加速されます。この結果，加速された電子がさらに別の電子ーイオンのペアを作り，さらに作られた電子がほかの原子を叩く，「電子なだれ」現象が起こることになります。

このため，最終的に電極にたどりつく電子やイオンの数は1次電子の数よりも多くなります。これは電極間の電圧を一定にしておけば1次電荷の数と比例します。つまり，天然増幅器になるわけです。これが比例計数管です。

さらに調子にのって電圧を上げていくと電子なだれがどんどん激しくなっていき，ついに，少しでも1次電子ができれば，電極全面にわたって電子なだれが起こるようになり，きわめて大きな出力が取り出せることになります。デジタル的な動きといえばよいのでしょうか。これがガイガー・ミューラー計数管(GM管と略されます)です。

ただ，このままにしておくと，(質量が大きいために）ゆっくり移動したイオンが陰極で電子と結合して紫外線が放出され，これがまた電離作用をしてしまうといったことになり，電子なだれがいつまでも止まらなくなってしまいます。そのため，ガイガー・ミューラー計数管では中にメタノールなどの多原子ガスやハロゲンガスを消滅剤として入れてあります。

電極間にかけている電圧は，電離箱で数10から200Vくらい，比例計数管では200から600V，ガイガー・ミューラー計数管で1000V程度です。

○半導体検出器

半導体検出器はいってみれば固体電離箱で，放射線電離作用で生成された電子ー正孔のペアを電極に集めてしまおうというものです。半導体のp-n接合面に逆電圧をかけると電流は流れませんが，ここに放射線が飛び込み，電子ー正孔のペアができると電流が流れます。

同じような現象を光で応用したのはフォトダイオードやフォトトランジスタで，こちらは赤外線リモコンなどでもお馴染みですね。半導体検出器は高電圧がいらないこと，高速応答性や検出率の点でもなかなかよい性能であることなどから，電離箱にとって代わった感があります。

◎メモ3:秋月キット

以前の秋月キットはかなりマニアックで，説明書とはまったく違う部品と入れ替わっていたり（電気的には置き換えられるとはいっても，形状などはまったく違うので，自分でピン接続を調べたり，あちこち削ったりしなければならなかった)，部品の過不足がすごかったり，基板のパターンは間違いだらけといったぐあいで，とても他人に勧められるようなものではありませんでしたが，最近はまともになったようです。今回のキットでも，部品は説明書どおりのまともなものが少し余分（壊してしまったときのことを考えたのだそうです）に入っていますし，基板や回路図の間違いなどもきちんと訂正の紙が入っています。実際に作ってみても，まあ，合格というレベルでした。秋月は通販もやっています。小さい店ですが秋葉原に店を構えて何十年，秋葉原でも1，2を争う有名どころですから通販体制もかなりしっかりしているほうだと思います。

ガイガーカウンタの製作

ここで終わってしまうのではちょっと寂しいので，例によって秋葉原をうろうろしてみました。さすがに放射線検出器などというものは，そうそう簡単に転がっているものではないようです。どうしようもなければ，霧箱でも作ってみるかと思っていたら，秋月電子通商（通販は〒158 世田谷区瀬田5-35-6：秋月通商：代金は現金書留か郵便為替：質問は往復葉書）に4,700円（通販のときは600円を加算：トランジスタ技術10月号広告より）でガイガー検出器のキットが出ていました。入手の面倒な計数管，高圧トランス（電流はほとんど流れないのでごく小さなものですが）に発振回路などの部品，ブザーにガラスエポキシの基板，アクリルケースまでついていますから部品集めの苦労はしなくてすみそうです。

このキットでは計測されたときに，発振器の出力をONして，ピッ！ と音が出るようにしています。この出力ON/OFF信号をカウントしてやれば，ガイガーカウンタになるわけです。さらに4,000円くらい出せば秋月でもLCD（液晶）表示のカウンタとペアにしたものもあります。

しかし，ただカウントするしか能がないのでは面白くありません。私たちは「電脳」という，強い仲間を持っているのですから，これを活用しない手はないでしょう。幸い，このキットの動作電圧は5～9Vですからちょっと手を加えればいつもの調子でジョイスティックポートにつなげられそうです。試しに5Vで動かしてみましたが，ブザーの音がちょっと小さくなるくらいで，特に問題はないようです。

これで，メドはたったのであとはジョイスティックポートとのインタフェイスを考えればよさそうです。簡単にやるなら，カウント用の出力をそのままジョイスティックポートに放り込んで，CPUの全力疾走で取り込むというやり方ですが，今回使ったガイガー・ミューラー計数管はかなり小型のものですから，通常検出される放射線(バックグラウンド放射線）は1分間に1～3発程度です。

この程度のパルスのためにCPUを全力疾走させるのはいくらなんでももったいないので，カウンタICを1個追加して，6ビットデータとして読み出せるようにしました。6ビットなら64パルスくるまではひと回りしませんから，1分に1回読みにいくだけで十分でしょう。

もし，1分に64パルスを超えてしまったら……そのときは悠長にこんな製作記事を読んでいないで，さっさと逃げ出したほうが身のためです。

○部品集め

というわけで，今回は市販キットに手を加えるということにしましたので，部品集めはいつもよりかなり楽です。

買い足さなくてはならない部品は次のとおりです。

9ピンD-SUBコネクタ（メス）	1
74HC393	1
0.1μFのセラミックコンデンサ	1
33kΩの抵抗	1

基板（ごく小さい物でよい）　1
10芯フラットケーブル（長さは適当）
ネジ，ナット（M３）　3組

基板は74HC393が乗ればいいだけなので，なにかを作ったときの切れ端でも十分です。なければ宙吊りでもかまわないでしょう。74HC393がなければ74LS393でもかまいません。

これらの部品はヒロセムセンパーツセンター（〒101 千代田区外神田1-10-5，☎225-2211内線255，284）などで手に入ります。通販を利用するときには電話で在庫，値段を確認して，送料1,000円を同封して現金書留で注文してくださいとのことです（トランジスタ技術10月号広告より）。

キットの値段のほうも10月現在ですので，注文する前には念のため雑誌の広告などを確認するようにしてください。

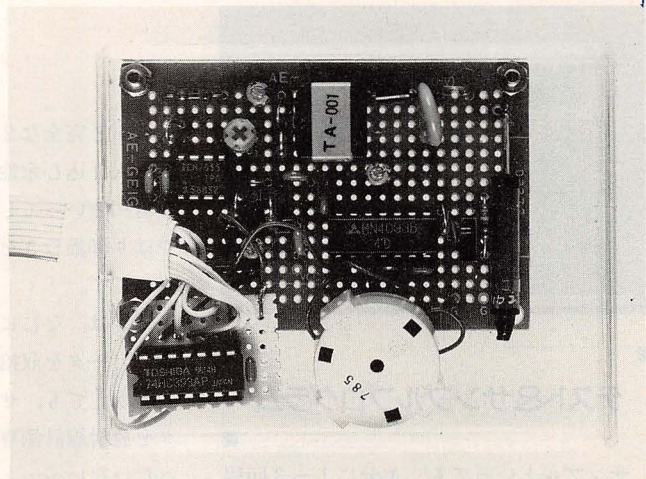
製作

キットのは006P（9Vの電池）を使うことを想定しているため，3端子レギュレータ（S-81350）を使って5V系の電源を作っていますが，今回は電源は本体からもらう5V単一で動かすので最終的にはこのレギュレータは不要になります。006Pの電池スナップとも，あとではずしやすいようにしておいてください。それ以外はキットの説明書どおりに組み立てます。片面基板のためにパターンが通りきらず，電線でつながなくてはならない部分がいくつかありますので，見落とさないように注意してください。

組み上がったら006Pの電池をつないでみます。しばらく（1～2分）おいておくと「ピッ」と小さな音がするはずです。だいたい1分に1～3回くらい鳴ると思います。

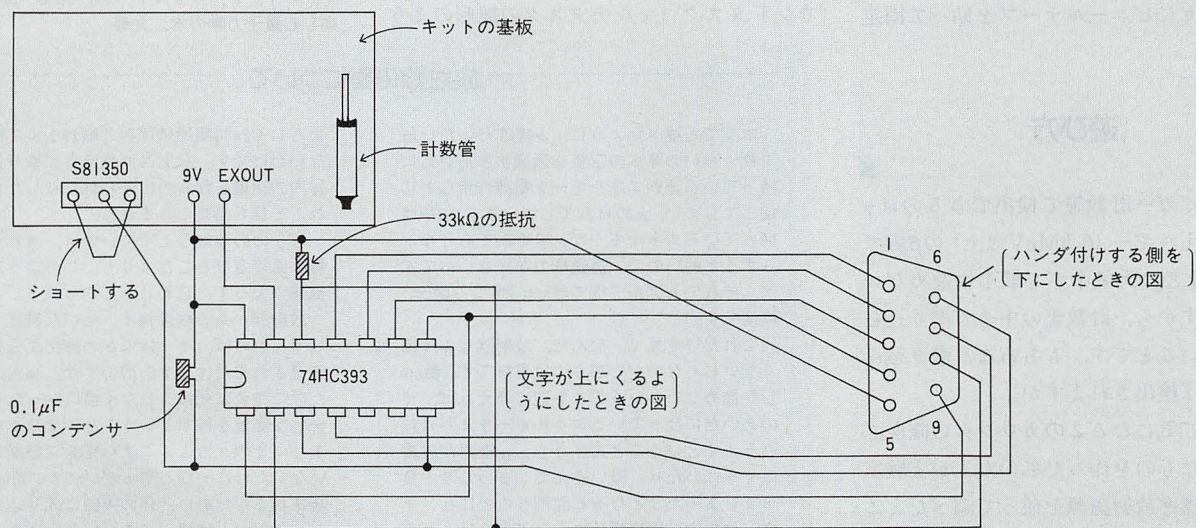
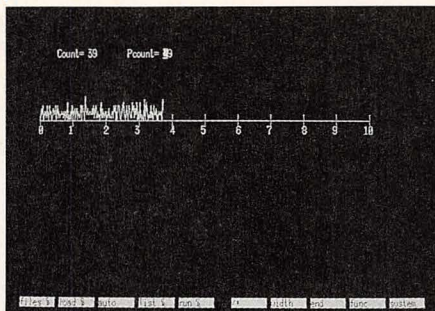
ここまでうまくいったら，次に改造とジョイスティックポートとの接続にかかります。まず，先ほど仮止めにしていた3端子レギュレータをはずし，3個の穴の両端をショートします。これで，電源が共通になります。

追加する74HC393は，本体の基板に入れるスペースがないので外付けになります。配線は図2を見てもらうことにしましょう。

9ピンのD-SUBコネクタとの接続が終わったら，配線のチェックをしておきましょう。D-SUBコネクタをよく見ると，ピン番号が書いてあるはずですから，ピン番号を逆に数えたりしないように注意してください。電源ピンの配線は特に念入りにチェックしてください。

テスターなどを持っていれば電源のショートなどがないかも念のためにチェックしておきましょう。

動作チェック

ジョイスティックコネクタに差し込み，電源を入れたまましばらく様子を見ていると，電池で動かしていたときと同じくらいの頻度で音がするはずです。電池のときよりも電圧が下がっている分，音も小さくなりますので，耳をそばだてて聞いてください。

図2　ガイガーカウンタ回路図

テスト&サンプルプログラム

サンプルといっても，1分に1～3回程度発生するパルスをカウントするだけですから，ワイヤレスリモコンのときのように波形がさっと表れるというわけにもいきません。データの収集は実に地道な作業です。

とりあえず，10時間データを収集しながら表示するようにしました。家に帰ってきてからスタートをかければ，明け方には終わっているでしょう。外部関数の入力もお忘れなく。

3分くらい，プログラムを動かしながらピッと鳴る回数を数えます。表示される回数と，音が鳴った回数は同じになるはずです。もしカウンタが2倍近い値をとるようなら，基板上の半固定抵抗をドライバなどで回して音がするたびにカウントが1だけ増えるように調整します。

うまくいっているようだったら，ケースの中に組み込みましょう。私のは，ちょっとふたの締まりが悪く，すぐ開いてしまうのでまわりにビニールテープを貼って固定しました。

遊び方

このガイガー計数管で検出できるのはγ線と高エネルギー（0.5MeV以上）のβ線です。α線などは大気中でも紙1枚進めないくらいですから，計数管の中まで潜り込むのも難しいことです。もちろん，潜り込んでくれれば検出されますが。

ところで気になるこのカウンタの確度ですが，同じものを作った私の友人が実験室にあった標準放射線源を使って調べたところ，かなり正確であるという評価をしていました。ガイガー計数管の場合には出力はほとんど完全な2値信号になるので，ノイズの入り込む余地もほとんどありませんから当然といってしまえば当然のことですが，やはり御墨つきをもらったようでうれしいものです。

あとは，なにに使うかです。やはり長期的にデータを収拾するのが面白い使い方でしょう。でも，せっかく買ったコンピュータを放射線計測専用にするのはちょっともったいない……と思った方は，雨水の測定なんていうのはいかがでしょうか。

降り始めの雨水は大気中のいろいろな浮遊物を付着させて落ちてきますから，妙なものが飛んでいれば雨水の中に凝縮されてきています。雨が降り始めたらすぐにお皿などに収集を行い，集めた雨水をそっと蒸発させて，お皿から一定の距離に置いたガイガーカウンタで1時間程度測定を行います。そのときのカウント値をメモしておくと，どこぞの国(日本国内かもしれません)で$E=mc^2$したかどうかわかるかもしれません。乳製品などには，環境の放射性同位元素も凝縮されますから，輸入ものがあったら面白半分に測ってみるというのもよいでしょう。いずれの場合にも被測定物との距離と，測定する時間を一定にするようにしないと，ほとんど意味のない数値が出てしまいます。

Humanもバージョン2.0で，バックグラウンドタスク（マルチタスクの雛形のようなものです）をサポートするようになったようですから，1分おきに起動してデータ収集をやらせておくようなプログラムを作っておくのも面白いでしょう。リボンをつければクリスマスプレゼントにもなりますし。

おわりに

ハードはやりたいけど，部品が……という方が意外に多かったので，今回は市販のキットをベースにして，ICをひとつだけ追加してX68000と接続してみました。デバイス（ガイガー計数管など）が手に入れにくいために，パソコン雑誌でもあまり取り上げられることのない周辺機器（?）ですが，コンピュータとペアになることでまたいろいろと使い道が考えられそうです（……しばし，沈黙)。

それにしても，テストプログラムのチェックをかねて48時間の連続運転をしたときに，6畳1間で夜中にかすかに響く「ピッ」の音の不気味なこと。たまに2発連続で「ピッ，ピッ」などとくると，確率の問題だとわかってはいてもやはり心臓によくありませんねぇ。今度やるときはもう少し平和に暮らせそうなものを作ろうと心に決めた私でした。

参考文献

T．ヘイ，P．ウォルターズ，大場一郎訳，目で楽しむ量子力学の本，丸善

放射線の害について

本文でも述べたように，α線はだいたい紙1枚くらいの厚さの空気を通過するあいだに持っている運動エネルギーを電離作用などに使ってしまい，止められてしまいます。β線は持っているエネルギーがα線並みでありながらずっと軽いので，透過能力が大きいのですが，それでもアルミ板で数mm（水なら数cm）程度です。

これだけを見て，なんだ，放射線なんて怖くないじゃないかと思うのは早計です。数cmしか進めないということを逆に考えれば，そのあいだに持っているエネルギーをまわりにバラまくということですから，α線源が皮膚にくっついたり，吸い込んでしまったり，食べてしまったりとなると面倒なことになります。体内の粘膜に張りつくと簡単にはとれませんから，そこから数cmのところはリ◯インよろしく24時間連続照射と戦わなくてはならないわけです。放射性同位元素が取り込まれ，体内の組織を作るのに使われたりしたら，それこそ目も当てられません。

原子力の事故などで怖いのは，洩れた放射線を直接浴びることよりも，このような内部被爆によって，じわじわとやられることです。

放射線がある程度強く，浴びた細胞が死滅してくれれば，そこはほかの細胞が分裂して修復されるだけですむのですが，適当に弱いために完全に死ぬことなく癌になったり，遺伝的な影響を残すということになります。いまだにそのメカニズムすら完全に把握できていないだけに一段とやっかいです。数cmで遮蔽されるがために，体の内部に入り込まれたら，外部からは検出できないということでもありますし。

リスト1

```
1000 int time
1010 int count,pcount,dcount,pdcount
1020 int hour,min,pmin
1030 screen 2,0,1,1
1040 init_screen()
1050 wait_a_minute()
1060 photorst()
1070 wait_a_minute()
1080 pcount=photoget():pdcount=pcount
1090 for time=0 to 599
1100    wait_a_minute()
1110    count=photoget()
1120    locate 10,5:print"Count=           Pcount="
1130    locate 16,5:print count:locate 33,5:print pcount
1140    if (count<pcount) then {
1150        dcount = 64-pcount+count
1160    } else {
1170        dcount = count-pcount
1180    }
1190    line(time+50, 200-pdcount*5, time+51, 200-dcount*5,
 15)
1200    pcount=count:pdcount=dcount
1210 next
1220 end
1230 func init_screen()
1240   int i
1250   line(50,200,650,200,11)
1260   for i=0 to 10
1270        line(i*60+50,190, i*60+50, 210,9)
1280   next
1290   locate 0,13
1300   print"     0       1       2       3       4       5
6       7       8       9      10"
1310 endfunc
1320 func get_time()
1330    str s
1340    s = time$
1350    hour = val(left$(s,2))
1360    min  = val(mid$(s,4,2))
1370 endfunc
1380 func wait_a_minute()
1390    repeat
1400        get_time()
1410    until(pmin<>min)
1420    pmin=min
1430 endfunc
```

リスト2

```
==================  PHOTON.S  ==================
  1: *-----------------------------------------------
  2: *- photo.s
  3: *-
  4: *- ガイガーカウンターテスト用外部関数
  5: *-
  6: *-      photorst:カウンターをリセットします
  7: *-      photoget:現在のカウンタの値を読みます
  8: *-----------------------------------------------
  9:         .include        doscall.mac
 10:         .include        fdef.h
 11:         .globl          _photoget
 12:         .globl          _photorst
 13:
 14: IOCS            equ     $0f
 15:
 16: PPI_PORT_A      equ     $e9a001
 17: PPI_PORT_B      equ     $e9a003
 18: PPI_PORT_C      equ     $e9a005
 19: PPI_CWR         equ     $e9a007
 20:
 21: PC4_ASSERT      equ     $8
 22: PC4_NEGATE      equ     $9
 23:
 24:         .text
 25:         .even
 26: *
 27: * インフォメーション・テーブル
 28: *
 29: *
 30:         .dc.l           AS_INIT
 31:         .dc.l           AS_RUN
 32:         .dc.l           AS_END
 33:         .dc.l           AS_SYS
 34:         .dc.l           AS_BRK
 35:         .dc.l           AS_CTRL_D
 36:         .dc.l           AS_RES1
 37:         .dc.l           AS_RES2
 38:         .dc.l           PTR_TOKEN
 39:         .dc.l           PTR_PARAM
 40:         .dc.l           PTR_EXEC
 41:         .dc.l           0,0,0,0,0
 42:
 43: AS_RES1:
 44: AS_RES2:
 45: AS_END:
 46: AS_BRK:
 47: AS_CTRL_D:
 48: AS_INIT:
 49: AS_RUN:
 50: AS_SYS:
 51:         rts
 52:
 53: *
 54: * トークン・テーブル
 55: *
 56: PTR_TOKEN:
 57:         .dc.b           'photoget',0
 58:         .dc.b           'photorst',0
 59:         .dc.b           0
 60:         .even
 61: *
 62: * パラメータ・テーブル
 63: *
 64: PTR_PARAM:
 65:         .dc.l           PHOTOGET_PAR
 66:         .dc.l           PHOTORST_PAR
 67:
 68: *
 69: * パラメータIDテーブル
 70: *
 71: PHOTOGET_PAR:
 72:         .dc.w           int_ret
 73: PHOTORST_PAR:
 74:         .dc.w           void_ret
 75:
 76: *
 77: * 関数アドレステーブル
 78: *
 79: PTR_EXEC:
 80:         .dc.l           photoget
 81:         .dc.l           _photorst
 82:
 83: *
 84: * スタック・バッファ
 85: *
 86: SPBUF:
 87:         .ds.l           1
 88:
 89:         .even
 90: *
 91: *
 92: *
 93: photoget:
 94:         bsr             _photoget
 95:         lea.l           PH_RETVAL,a0
 96:         movea.l         a0,a1                   * おまじない
 97:         move.w          #0,(a0)
 98:         move.l          #0,2(a0)
 99:         move.l          d0,6(a0)
100:         moveq.l         #0,d0
101:         rts
102:
103: _photoget:
104:         clr.l           -(sp)           *    SPBUF = _SUPER(0);
105:         dc.w            _SUPER
106:         addq.l          #4,sp
107:         move.l          d0,SPBUF
108:
109:         move.b          PPI_PORT_A,d2
110:         move.b          d2,d1
111:         andi.l          #$f,d2
112:         andi.b          #$60,d1
113:         lsr.b           #1,d1
114:         add.b           d1,d2
115:
116:         move.l          SPBUF,d1
117:         bmi             photoget_already_super
118:         move.l          d1,-(sp)
119:         dc.l            _SUPER
120:         addq.l          #4,sp
121: photoget_already_super:
122:         move.l          d2,d0
123:         rts
124:
125: *
126: *
127: _photorst:
128:         clr.l           -(sp)           *  SPBUF = _SUPER(0);
129:         dc.w            _SUPER
130:         addq.l          #4,sp
131:         move.l          d0,SPBUF
132:         movea.l         #PPI_CWR,a0     *  *ppi_cwr = RQ_NEGATE;
133:         move.b          #PC4_NEGATE,(a0)
134:         nop
135:         nop
136:         move.b          #PC4_ASSERT,(a0) * *ppi_cwr = RQ_ASSERT;
137:         move.l          SPBUF,d1        * _SUPER(SPBUF);
138:         bmi             photorst_already_super
139:         move.l          d1,-(sp)
140:         dc.l            _SUPER
141:         addq.l          #4,sp
142: photorst_already_super:
143:         moveq.l         #0,d0
144:         lea.l           PH_RETVAL,a0
145:         movea.l         a0,a1
146:         move.w          d0,2(a0)
147:         rts
148:
149: *
150: PH_RETVAL:
151:         .dc.w           0
152:         .dc.w           0
153:         .dc.w           0
154:         .dc.w           0
155:         .dc.w           0
156:         .end
```

Oh!X 2周年記念

特別モニタ&愛読者プレゼント

いやー，めでたいめでたい。なんとOh!Xも2周年を迎えることができました。これもひとえに皆様のおかげです。ありがたやありがたや。それにしても月日のたつのは早いもんですねえ。1988年12月号のあの大プレゼント大会からはや1年。今年もこの季節がやってまいりました。去年にも増してたくさんのプレゼントの数々。皆様に日頃の感謝の気持ちをこめて，「もってけドロボー！」

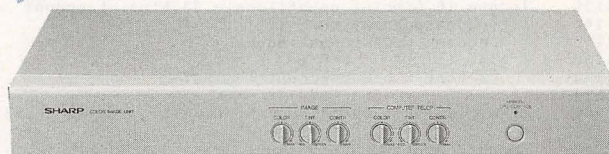
ハードウエア

1

X68000 PRO

CZ-652C+CZ-603D　合計価格382,800円　1名

ディスプレイと本体のセット。X1ユーザーや MZ ユーザーのキミ，奮って応募しよう。

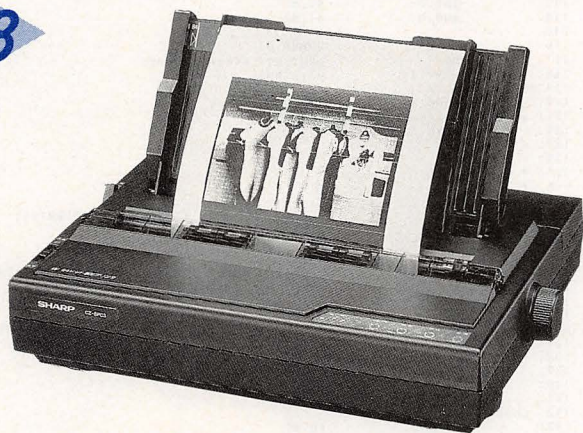
2

カラーイメージユニット

CZ-6VT1　69,800円　1名

ビデオにつないで遊べればなぁ，と思っていた方，この1台でその夢が実現できちゃうのです。

特別モニタプレゼント
（シャープ提供）

今年もOh!Xの2周年を記念して，シャープさんからたくさんのモニタプレゼントのご提供をいただきました。今年の目玉商品は，な，なんとX68000 PROをディスプレイとセットでプレゼント！ X68000を横目で見ながら指をくわえていた方々，この機を逃してはいけませんぞ。そのほかカラーイメージユニットやプリンタなど，ぜひ欲しいと思うものを用意してくださいました。さすがユーザーの心理がわかってらっしゃる！

3

熱転写カラープリンタ

CZ-8PC3　65,800円　1名

本体とディスプレイは持っているけど，プリンタにまで手が回らなかった人って意外と多いのでは……？

4

サイバースティック

CZ-8NJ2　23,800円　3名

ゲーマーには必携のサイバースティック。これがあれば鬼に金棒，天下無敵のジョイスティックだ。

5

トラックボール

CZ-8NT1　13,800円

3名

パッドやジョイスティックに慣れ親しんでいるとはいえ，たまにはコロコロとトラックボールで遊んでみたいよね。

6

Ccompiler PRO-68K

CZ-211LS

39,800円　1名

X-BASICならもうカンペキ，というあなた，こんどはCに挑戦していろいろなプログラムを組んでみましょう。

7

OS-9/X68000

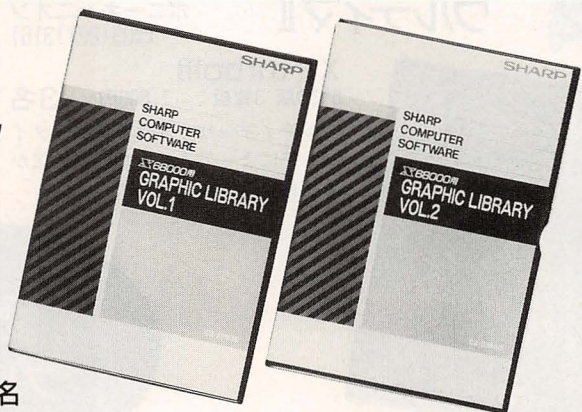
CZ-219SS　29,800円　1名

高性能なオペレーションシステムは，プログラマを助ける？ X68000の機能をフルサポートしてくれます。

8

グラフィックライブラリ VOL.1&VOL.2

CZ-235GS, CZ-236GS

8,800円　3名

暑中見舞いやクリスマス，はたまた年賀状にも利用できるアートツールセットです。

ソフトウェア

9

ソングライブラリ（101曲集）

CZ-248MS　8,800円　3名

MUSIC PRO-68KとMUSIC PRO-68K MIDI用の曲がギッシリ詰まったデータ曲集。バックミュージックにいかが？

10

ゲームソフト「お好きなゲーム1本プレゼント」

5名

a. パックマニア
b. ツインビー
c. 沙羅曼蛇
d. 熱血高校ドッジボール部
e. アルカノイド
f. ニュージーランドストーリー
g. フルスロットル

上記のゲームソフトの中から，好きなソフトをどれかひとつだけプレゼント。希望するソフトを明記のこと（例：10-a）。

プレゼントの応募方法

とじ込みのアンケートはがき（ただし今月号のもの）の該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1989年12月18日の到着分までとします。当選の発表は1990年２月号で行います。

維新の嵐

光栄　☎044(61)6861

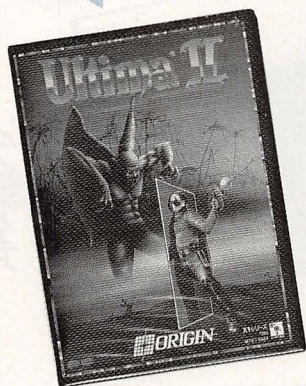
X1turbo用 5″2D版 3枚組 9,800円 3名

激動の明治時代を生き抜いてきた英雄たちとその思想を描いたシミュレーションゲーム。

12 ウルティマII

ポニーキャニオン
☎03(221)3161

X1/turbo用
5″2D版 3枚組　7,800円　3名

ウルティマシリーズの第２弾。タイムドアによってまたまた新しい広がりを見せてくれたRPGだ。

SUPER DEVICE MONITOR "T"

BLUE SKY
☎0559(72)6710

X68000用
5″2HD版　15,000円
2名

これひとつでプログラムの開発がラクになるぞ。

愛読者プレゼント

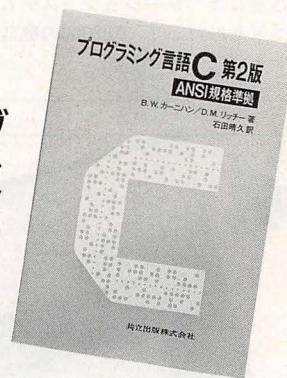
14 DIGITAL IC TRAINER

サンハヤト　☎03(984)7719
4,950円　1名

楽しくデジタル回路が学べるというデジタルIC実験キット。ガイドブックつき。

プログラミング言語C 第2版

共立出版　☎03(947)2511
2,800円　2名

ちょっと難しいかもしれないけど，持っていてソンはない本だと思うよ。

16 清涼飲料水　1名

ROOT BEERは石川県の小森君から，キビジュースはモニタの湯澤さんからのプレゼント。毒は入ってません。

10月号プレゼント当選者

1高速マルチスクリーンエディタJAMES68K（東京都）田中聡（神奈川県）林弘和　2ソーサリアン・宇宙からの訪問者（静岡県）高倉真樹（大阪府）山崎聡士（広島県）久森真介　3ガウディ・バルセロナの風（栃木県）柴山一男（埼玉県）垣矢信行（愛知県）枡田浩義　4麻雀狂時代SPECIAL II・冒険編（大阪府）光井浄二（兵庫県）七浦啓有　末吉宏　5ねじ式（東京都）菅原巌（栃木県）梶原康延（高知県）狩野信雄　（敬称略）

以上の方々が当選されました。おめでとうございます。品物は順次発送いたしますが，入荷状況などにより遅れることがあります。また，公正取引委員会の告示により，このプレゼントに当選された方は，この号の他の懸賞には当選できない場合がありますのでご了承ください。

タートルグラフィックの話

Izumi Daisuke 泉　大介

今回は三角関数を使ってタートルグラフィックに挑戦してみましょう。タートル関数ができたら，それを使ってさまざまな再帰図形を描くことができます。再帰処理はX-BASICの特徴のひとつです。この機会に考え方に慣れておいてください。

先月は皆さんのご要望にお応えして，グラフィック画面のスクロールをやりました。本来なら大変面倒な処理となるグラフィック画面のスクロールも，X68000のハードウェアの力のおかげで簡単に実行できることがおわかりいただけたかと思います。今回はLOGOやSmalltalkといった言語で採用されているタートルグラフィックをやってみたいと思います。高校の数学で習う三角関数の格好の例題でもありますので，つまずいてしまった皆さんはここで復習してみてください。

タートルグラフィックとは

タートルとは「亀」という意味です。尻尾に筆をつけた亀を想像してみてください。この亀を紙の上に置き，「おら，前に進め！」，「おら，左を向け！」，などと命令して絵を描かせようというのがタートルグラフィックです。四角形を描くのに，X-BASICでは「左上の座標と右下の座標を決め，box関数を使って」描きます。タートルグラフィックでは「一定距離進んで90°右を向くことを4回繰り返して」描くことになります。

一見，図形を描くのが繁雑になるような気がするかもしれません。確かに一般の座標系に慣れてしまった私たちにとっては痒いところに手が届かないような方法ですが，角度を習ったばかりの子供が図形を描くには実に自然なやり方であり，LOGOのような入門用の言語で採用されている理由もここにあります。正三角形を描く場合，タートルグラフィックでは「一定距離進んで120°右を向くことを3回繰り返せばいい」のに対し，一般の座標系では3点を決めるのにちょっとした計算を必要とすることからも，タートルグラフィックの簡易性がおわかりいただけるかと思います。

一般の座標系に近い方法で図形を描くコンピュータ画面（上下が逆になっているだけ）にタートルグラフィックを実現するには，亀の動きを座標で表現する作業が必要となります。では，その方法を考えていきましょう。

三角関数とグラフィック

習ったときには，いったい何に使えるのか見当もつかないのが高校でやる数学です。三角関数など，直角三角形の辺の長さを求める以外にどんな使い方があるのかと怪しんだものです。特に内積は，2つの直線が直交するかどうかを調べるためにあるものだと決めてかかっていました。内積の実においしい利用法を見つけたのは最近のことです。これはまた，別のテーマでお届けすることにしましょう。

sin，cosでお馴染みの三角関数は，図1のように定義されています。直角三角形の斜辺の長さで，それぞれの短辺を割ったものです。したがって原点から点aまでの長さを1とすると，図2における点aの座標は，$(\cos\theta,\ \sin\theta)$と表されます。亀が角度$\theta$の方向を向いて距離100移動するならば，移動後の

$\cos\theta=\frac{c}{a}$

$\sin\theta=\frac{b}{a}$

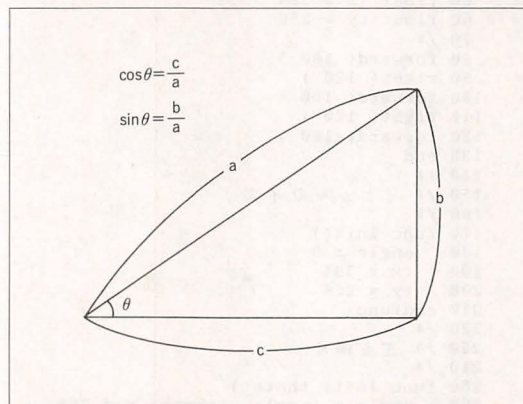



図1　三角関数の定義

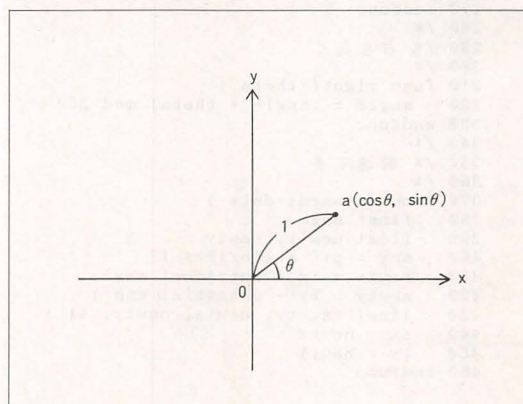



図2　点aの座標

座標は（100×cosθ，100×sinθ）になりますね。

次に点aを，さらに角度ηだけ回転させた場所に移動してみます（図3）。全体としてみれば角度θ＋ηになっていますから，新しい点a'の座標は（cos(θ＋η)，sin(θ＋η)）になります。亀がこの方向に距離100進んだあとの座標は（100×cos(θ＋η)，100×sin(θ＋η)）です。つまり，角度を保存しておく変数を用意すれば，亀が移動したあとの座標を簡単に求めることができるわけです。

亀は常に，現在自分がいる場所を座標原点と考えて移動後の座標を計算します。そこで，亀が現在いる座標を保存しておき，先の方法で計算した移動後の座標を加えてやれば，亀が実際にどこに移動するのか知ることができるわけです。

1） 一般の座標系では，反時計回りに回転すると回転角が増えます。

図3 さらにη回転させたa'の座標

リスト1 タートルグラフィック

```
 10 /* タートルグラフィック
 20 /*
 30 screen 2,0,1,1
 40 int   angle = 0
 50 float tx = 384
 60 float ty = 256
 70 /*
 80 forward( 100 )
 90 right( 120 )
100 forward( 100 )
110 right( 120 )
120 forward( 100 )
130 end
140 /*
150 /* イニシャライズ
160 /*
170 func init()
180   angle = 0
190   tx = 384
200   ty = 256
210 endfunc
220 /*
230 /* 左を向く
240 /*
250 func left( theta )
260   angle = (angle - theta) mod 360
270 endfunc
280 /*
290 /* 右を向く
300 /*
310 func right( theta )
320   angle = (angle + theta) mod 360
330 endfunc
340 /*
350 /* 前進する
360 /*
370 func forward( dots )
380   float ang
390   float newtx, newty
400   ang = pi( angle/180# )
410   newtx = tx + dots*cos( ang )
420   newty = ty + dots*sin( ang )
430   line( tx, ty, newtx, newty, 11 )
440   tx = newtx
450   ty = newty
460 endfunc
```

リスト1はこの方法をそのままプログラムにしたものです。40行で宣言している変数angleが亀の角度を意味します。続くtxとtyは亀のx座標とy座標です。プログラムは4つの関数からできています。

init関数は亀を初期化します。角度を0°に，亀の位置を画面中央に設定し終了です。

left関数とright関数は，亀に左右を向かせる関数です。グラフィック画面は一般の座標系の上下をひっくり返したように座標をとりますから，時計回りに回転したときに回転角が増えるようにすればつじつまが合います[1)]。亀が右を向けば回転角が増え，左を向けば回転角が減ります。260，320行の符号が異なっている点に注意してください。

残る前進はforward関数です。sin, cos関数はラジアン角の引数をとります。そこでpi関数を使って，angleをラジアン角に変換します。180°がπラジアンですから，angle÷180をpi関数に渡せばOKです。この角度を元に，新しい亀の座標を410，420行で計算し，430行で線を引きます。そして亀の現在位置を更新すれば終了です。

80～130行のプログラムは，例として正三角形を描いて終了します。プログラム中で定義した関数は，X-BASICが最初からもっている関数と同じように使えますから，画面にOKと表示されたあと，

wipe（）
init（）

とダイレクトモードで入力すればプログラム実行前と同じ状態になります。

forward(50)
right(60)

と入力したあと，カーソルをforward(50)と入力した行に戻し，リターンキーを2回押せば次の辺が描けます。これを繰り返すと正六角形のできあがりです。タートルグラフィックのインタプリタのように使えますので試してみてください。

亀を表示する

このままでは，現在亀がどっちを向いているのかわかりづらいですね。そこで画面上に亀を表示することにしましょう。亀らしい形にすると表示に時間がかかりますから，単純な五角形（野球のホームベースのような形）とします。また，亀が図形を消してしまわないように，グラフィック画面を複数使い，亀を表示する画面と図形を表示する画面を分けることにします。複数の画面を使う方法は先月やりましたね。512×512の画面で，使う色数を16にすればグラフィック画面を4つ使えるようになりますし，使

う色数を256色とすれば２つ使えます。

亀は「30°左向け～左！」と命令されれば左を向きます。そうでなければ亀を表示する意味がありません。これは次のように２段構えで対処します。

1) 亀の真ん中を座標原点とし，回転させる
2) 回転させた亀を目的の座標へ移動させる

亀の真ん中を座標原点として原点回りに回転させると，亀を目的の方向に向けることができます。亀を表す五角形は原点回りに散らばった５つの点を結んだものとみなすことができますから，ある点を原点回りに回す方法がわかれば亀を回転できるわけです。ある点を原点回りに回す……そう，数学でやる回転行列を使えばいいですね。

回転行列は２×２の行列で，要素は$\cos\theta$と$\sin\theta$で構成されています。ここまではちょっとマジメに数学をやった人なら覚えていると思います。ただ，$\cos\theta$と$\sin\theta$がどのような順番で入っていたか，どっかに‘－’がついていたと思うけどどこについていたか，ということを正確にいえる人は現役バリバリの理工系の高校生くらいでしょう。いくら理工系でも，大学に入って２，３年もたてばもうだめです。こんなときは，実際に座標を入れてやってみるのがてっとり早い方法です。点（３，４）を90°回転させると，(－4，3) に移動します。つまり，

$$\begin{pmatrix}\alpha & \beta \\ \gamma & \delta\end{pmatrix}\begin{pmatrix}3 \\ 4\end{pmatrix}=\begin{pmatrix}-4 \\ 3\end{pmatrix}$$

ということですね。$\alpha\sim\delta$は$\cos\theta$か$\sin\theta$で，θは 90°ですから，$\alpha\sim\delta$は０か１のどちらかです。そしてどこかに‘－’がついていたことを考慮すると，

$$\begin{pmatrix}0 & -1 \\ 1 & 0\end{pmatrix}\begin{pmatrix}3 \\ 4\end{pmatrix}=\begin{pmatrix}-4 \\ 3\end{pmatrix}$$

となります。つまり角θだけ回転させる行列は，

$$\begin{pmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{pmatrix}$$

だとわかるわけです。$\cos\theta$に‘－’がつかないのかと疑問を持つ疑り深い方は，(3，4) を 180° 回転させてみれば確認できます。

この行列を使って亀を構成する５つの点を回転させれば，亀の回転はOKですね。ではプログラムにかかりましょう。リスト２です。

10行は画面の設定です。512×512ドットで16色使うことにしました。txとtyはリスト１と同じく亀の座標を入れます。次のAが回転行列，Ｔには亀を表す５つの点の座標が入ります。theta は亀を何度回転させるかを入力してもらうための変数です。

関数の説明をしておきましょう。setAという関数は，何度回転するかを引数に受け取り，その回転行列を配列Ａに作る関数です。trans関数は，回転行列を使って回転後の５つの座標を決定します。ちょっと込み入っていますが，たとえば，300，310行は，

$$\begin{pmatrix}T(0,\ 0) \\ T(0,\ 1)\end{pmatrix}=\begin{pmatrix}A(0,\ 0) & A(0,\ 1) \\ A(1,\ 0) & A(1,\ 1)\end{pmatrix}\begin{pmatrix}10 \\ 0\end{pmatrix}$$

を計算しています。つまり (10，0) を回転させた座標を計算しているのです。最後に0.5を加えているのは，配列Ｔが整数の配列だからです。整数だと，たとえ計算結果が2.9でも２が代入されてしまいます。0.5を加えておけば，2.5～3.4は３になりますから，「四捨五入」できるわけです。最後にprintT関数は亀を表示する関数です。亀が図形に隠れて見えなくなってしまうのを避けるため，亀を第０ページに，図形を第１ページに描くことにしました。trans関数で計算した５つの座標に亀の現在位置を加え，順に線で結んで亀を表示します。表示が終わったら，描画ページを再び第１ページに戻します。

プログラムは，まず70，80行で亀の位置を設定します。続く90～140行がwhile～endwhileのループを作っています。条件が１になっていますので，このループは無限ループです。100行に見慣れない命令が

リスト２　タートルを表示する

```
 10 screen 1,1,1,1 /* 512×512×16色
 20 int   tx, ty   /* 亀の座標
 30 float A(1,1)   /* 回転行列
 40 int   T(4,1)   /* 亀の５つの点
 50 int   theta    /* 亀の回転角
 60 /*
 70 tx = 256
 80 ty = 256
 90 while 1
100   input "回転角は ", theta
110   setA( theta )
120   trans()
130   printT()
140 endwhile
150 end
160 /*
170 /* 回転行列を決める
180 /*
190 func setA( theta;float )
200   theta = pi( theta/180 )
210   A(0,0) = cos( theta )
220   A(0,1) = -sin( theta )
230   A(1,0) = sin( theta )
240   A(1,1) = cos( theta )
250 endfunc
260 /*
270 /* 回転させる
280 /*
290 func trans()
300   T(0,0) = A(0,0)*10 + A(0,1)*0 + 0.5 /* (10,0) */
310   T(0,1) = A(1,0)*10 + A(1,1)*0 + 0.5
320   T(1,0) = A(0,0)*5  - A(0,1)*5 + 0.5 /* (5,-5) */
330   T(1,1) = A(1,0)*5  - A(1,1)*5 + 0.5
340   T(2,0) = -A(0,0)*5 - A(0,1)*5 + 0.5 /* (-5,-5) */
350   T(2,1) = -A(1,0)*5 - A(1,1)*5 + 0.5
360   T(3,0) = -A(0,0)*5 + A(0,1)*5 + 0.5 /* (-5,5) */
370   T(3,1) = -A(1,0)*5 + A(1,1)*5 + 0.5
380   T(4,0) = A(0,0)*5  + A(0,1)*5 + 0.5 /* (5,5) */
390   T(4,1) = A(1,0)*5  + A(1,1)*5 + 0.5
400 endfunc
410 /*
420 /* 亀を表示する
430 /*
440 func printT()
450   int i
460   apage(0)
470   fill( tx-10, ty-10, tx+10, ty+10, 0 )
480   for i=0 to 3
490     line( T(i,0)+tx, T(i,1)+ty, T(i+1,0)+tx, T(i+1,1)+ty,
 9 )
500   next
510   line( T(4,0)+tx, T(4,1)+ty, T(0,0)+tx, T(0,1)+ty, 9 )
520   apage(1)
530 endfunc
```

ありますね。これは，プログラム中で変数にキーボードから値を入れるために用意されている命令です。

```
int   a
input a
```

とすると，画面に「？」が表示されてカーソルが点滅します。適当な数字をキーボードから入力しリターンキーを押すと，数字が数値に変換され，変数aにセットされます。

```
print  a
```

として確認してみてください。100行のように使えば，画面に「回転角は」と表示されたのちカーソルが点滅します。何を入れればいいのかわかりやすく便利ですね。input命令は数値だけでなく，文字列も受け取ることができます。このときは，

```
str   s
input s
```

とすればOKです。input命令の後ろにある変数によって，何を受け取るかが決まるわけです。回転角をキーボードから入力すると，先に説明した関数を順に呼び出し，亀が表示されます。回転角をいろいろと変えて試してみてください。

タートルグラフィック完成

リスト1とリスト2を合体させると，タートルグラフィックプログラムの完成です(リスト3)。リスト1，リスト2から抽出した関数はわかりやすいように1000行以降にまとめておきました。

回転行列の求め方

本文中では，忘れてしまったときの思い出し方という形で回転行列を紹介した。しかし，納得いかん！　という方のために，改めてここで点 (a, b) を角 θ だけ回転させた場合にどこに移動するのかを調べてみよう。x軸方向の単位ベクトルを $\overrightarrow{ex}$，y軸方向の単位ベクトルを $\overrightarrow{ey}$ とすると，点 (x, y) の位置ベクトルPは次のように書くことができる。

$$P=a\overrightarrow{ex}+b\overrightarrow{ey}$$

(a, b)という表現は，位置ベクトルの表現方法と同じであることを思い出してほしい。さて，この点(a, b)を角 θ だけ回転させると，図aのように(a′, b′)に移動する。ここで ϕ はx軸と点(a, b)のなす角である。このまま眺めていても点(a, b)をどうすれば(a′, b′)に移動できるかはわからない。視点を変えてみることにしよう。

図bを見ていただきたい。新しくx′軸とy′軸が設けてある。x′y′軸は，xy軸に対して角 θ だけ回転させた軸である。上と同じようにx軸方向の単位ベクトルを $\overrightarrow{ex'}$，y′軸方向の単位ベクトルを $\overrightarrow{ey'}$ とすると，

$$P'=a\overrightarrow{ex'}+b\overrightarrow{ey'}$$

と書ける。単位ベクトルが変わっただけで，それぞれの係数は変わらない。係数が変わらないわけは，図aと見比べてもらえれば明らかだろう。x′y′軸と点(a′, b′)の位置関係は，xy軸と点(a, b)の位置関係と同じである。

ということは，$\overrightarrow{ex'}$ ベクトルを $\overrightarrow{ex}$ ベクトルで，$\overrightarrow{ey'}$ ベクトルを $\overrightarrow{ey}$ ベクトルで書き表すことさえできれば，角 θ だけ回転させたあとの座標(a′, b′)を，$\overrightarrow{ex}$ ベクトルと $\overrightarrow{ey}$ ベクトルで書き表すことができるということである。$\overrightarrow{ex}$ ベクトルと $\overrightarrow{ey}$ ベクトルを角 θ だけ回転させると，単位ベクトルは長さ1のベクトルなので，

$$\overrightarrow{ex}:(1,\ 0)\ \rightarrow\ \overrightarrow{ex'}:(\cos\theta,\ \sin\theta)$$

$$\overrightarrow{ey}:(0,\ 1)\ \rightarrow\ \overrightarrow{ey'}:(-\sin\theta,\ \cos\theta)$$

と変換される(図c)。もうおわかりだろう。上の変換を行列を使って書き表すなら，

$$\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}$$

になるのである。この行列に $\overrightarrow{ex}$(1, 0)，$\overrightarrow{ey}$(0, 1)を掛け，実際に $\overrightarrow{ex'}$，$\overrightarrow{ey'}$ となることを確認してみてもらいたい。

(a, b)は「$a\overrightarrow{ex}+b\overrightarrow{ey}$」と書くことができた。上の回転行列をAと書けば，(a′, b′)は，

$$aA\overrightarrow{ex}+bA\overrightarrow{ey}$$

となる。回転してからa倍するのも，a倍してから回転するのも同じなので，上の式は，

$$A(a\overrightarrow{ex})+A(b\overrightarrow{ey})$$

としても構わない。さらに結合法則を使うと，

$$A(a\overrightarrow{ex}+b\overrightarrow{ey})$$

と表すことができる。「$a\overrightarrow{ex}+b\overrightarrow{ey}$」は，(a, b)と同じであるから，これは，

$$\begin{pmatrix}a'\\ b'\end{pmatrix}=\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}a\\ b\end{pmatrix}$$

と同じである。こうして点(a, b)を角 θ だけ回転させたあとの座標(a′, b′)を得ることができるわけである。

余談になるが，ここまでくると加法定理も90%終わったようなものなのでついでに紹介しておこう。簡単のため，点(a, b)と原点との距離を1とする。点(a, b)とx軸のなす角は，図aによると ϕ である。したがって，点(a, b)は点($\cos\phi$, $\sin\phi$) と書くこともでき，点(1, 0)を角 ϕ だけ回転させた点になっている。これを先の回転行列を使って，さらに θ だけ回転させてみよう。

$$\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}\cos\phi\\ \sin\phi\end{pmatrix}$$

実際に計算すると，

$$\cos\theta\cos\phi-\sin\theta\sin\phi$$

$$\sin\theta\cos\phi+\cos\theta\sin\phi$$

となる。計算して出てきた座標は，点(1, 0)を角 ϕ だけ回転させ，さらに角 θ だけ回転させたものなのだから，その座標は($\cos(\theta+\phi)$, $\sin(\theta+\phi)$)と等しい。よって，

$$\cos(\theta+\phi)=\cos\theta\cos\phi-\sin\theta\sin\phi$$

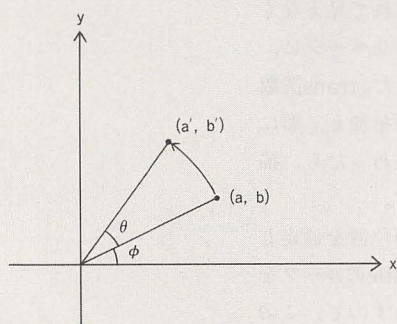
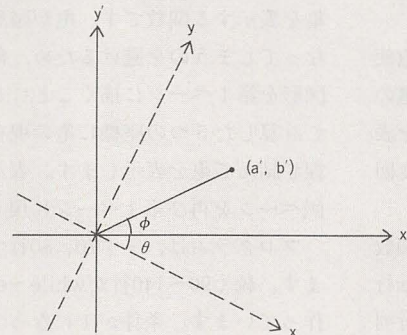
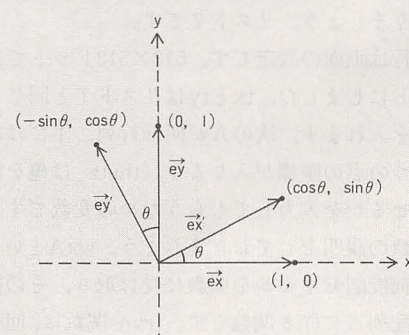
$$\sin(\theta+\phi)=\sin\theta\cos\phi+\cos\theta\sin\phi$$

がいえる。これが加法定理である。

sin(30°)やcos(45°) は覚えていても (覚えている必要があるが)，sin(15°)など誰も覚えてなどいない。このようなときも

$$\sin(15^\circ)=\sin(45^\circ-30^\circ)$$

と分解すれば上の加法定理から求めることができるわけである。この加法定理の求め方はそのまま覚え方として通用する。受験生の皆さん，頑張っていただきたい。

図a　(a,b)を θ だけ回転

図b　x′y′軸を導入

図c　単位ベクトルの回転

リストには若干の変更が加えてあります。リスト1では亀の座標を実数型の変数に格納していましたが，ここでは整数型とし，前進するたびに先ほど説明した「四捨五入」することにしました。1420，1430行です。また亀を表示するか，表示しないかを選択できるようにしました。亀を回転させるのにちょっと時間がかかるためです。

dispT（0）

とすると亀が表示されなくなり，0以外のパラメータを使うと亀が表示されるようになります。これに合わせ，printT関数にも変更が加えられています。

また，ペンの上げ下げができるよう，penUp, penDownの2つの関数もつけておきました。

リスト1，リスト2を利用してリスト3を作るには，まず，リスト1をload命令で読み込み，

save@“t1”

で保存します。save@は行番号をつけないでプログラムを保存する命令です。次にリスト2を読み込み，

renum 1000

として，行番号を1000からに変更します。そして，

load@“t1”

とすれば，あら不思議，1000行以降のリスト2は元

リスト3
亀つきタートルグラフィック

```
  10 /*
  20 /* 亀つきタートルグラフィック
  30 /*
  40 int   tx, ty     /* 亀の座標
  50 float A(1,1)     /* 回転行列
  60 int   T(4,1)     /* 亀の5つの点
  70 int   angle      /* 亀の回転角
  80 int   TurtleOff /* 亀の表示フラグ
  90 int   PenDown    /* ペンフラグ
 100 /*
 110 int i
 120 init()
 130 for i=1 to 3
 140   forward(100)
 150   right(120)
 160 next
 170 for i=1 to 4
 180   forward(100)
 190   right(90)
 200 next
 210 for i=1 to 5
 220   forward(100)
 230   right(72)
 240 next
 250 for i=1 to 6
 260   forward(100)
 270   right(60)
 280 next
 290 end
1000 /*
1010 /* イニシャライズ
1020 /*
1030 func init()
1040   screen 1,1,1,1 /* 512×512×16色
1050   angle = 0
1060   tx = 256
1070   ty = 256
1080   apage(1)
1090   TurtleOff = 0
1100   PenDown = 1
1110   setA()
1120   trans()
1130   printT()
1140 endfunc
1150 /*
1160 /* 左を向く
1170 /*
1180 func left( theta )
1190   angle = (angle - theta) mod 360
1200   setA()
1210   trans()
1220   eraseT()
1230   printT()
1240 endfunc
1250 /*
1260 /* 右を向く
1270 /*
1280 func right( theta )
1290   angle = (angle + theta) mod 360
1300   setA()
1310   trans()
1320   eraseT()
1330   printT()
1340 endfunc
1350 /*
1360 /* 前進する
1370 /*
1380 func forward( dots )
1390   float ang
1400   int   newtx, newty
1410   ang = pi( angle/180# )
1420   newtx = tx + dots*cos( ang ) + 0.5
1430   newty = ty + dots*sin( ang ) + 0.5
1440   eraseT()
1450   if PenDown then line( tx, ty, newtx, newty, 11 )
1460   tx = newtx
1470   ty = newty
1480   printT()
1490 endfunc
1500 /*
1510 /* 回転行列を決める
1520 /*
1530 func setA()
1540   float theta
1550   theta = pi( angle/180# )
1560   A(0,0) = cos( theta )
1570   A(0,1) = -sin( theta )
1580   A(1,0) = sin( theta )
1590   A(1,1) = cos( theta )
1600 endfunc
1610 /*
1620 /* 回転させる
1630 /*
1640 func trans()
1650   T(0,0) = A(0,0)*10 + A(0,1)*0 + 0.5 /* (10,0) */
1660   T(0,1) = A(1,0)*10 + A(1,1)*0 + 0.5
1670   T(1,0) = A(0,0)*5  - A(0,1)*5 + 0.5 /* (5,-5) */
1680   T(1,1) = A(1,0)*5  - A(1,1)*5 + 0.5
1690   T(2,0) = -A(0,0)*5 - A(0,1)*5 + 0.5 /* (-5,-5) */
1700   T(2,1) = -A(1,0)*5 - A(1,1)*5 + 0.5
1710   T(3,0) = -A(0,0)*5 + A(0,1)*5 + 0.5 /* (-5,5) */
1720   T(3,1) = -A(1,0)*5 + A(1,1)*5 + 0.5
1730   T(4,0) = A(0,0)*5  + A(0,1)*5 + 0.5 /* (5,5) */
1740   T(4,1) = A(1,0)*5  + A(1,1)*5 + 0.5
1750 endfunc
1760 /*
1770 /* 亀を表示する
1780 /*
1790 func printT()
1800   int i
1810   if TurtleOff then return()
1820   apage(0)
1830   for i=0 to 3
1840     line( T(i,0)+tx, T(i,1)+ty, T(i+1,0)+tx, T(i+1,1)+ty,
9 )
1850   next
1860   line( T(4,0)+tx, T(4,1)+ty, T(0,0)+tx, T(0,1)+ty, 9 )
1870   apage(1)
1880 endfunc
1890 /*
1900 /* 亀を消す
1910 /*
1920 func eraseT()
1930   apage(0)
1940   fill( tx-10, ty-10, tx+10, ty+10, 0 )
1950   apage(1)
1960 endfunc
1970 /*
1980 /* 亀の表示スイッチ
1990 /*
2000 func dispT( flag )
2010   if flag then {
2020     TurtleOff = 0
2030     trans()
2040     printT()
2050   } else {
2060     TurtleOff = 1
2070     eraseT()
2080   }
2090 endfunc
2100 /*
2110 /* ペンの上下
2120 /*
2130 func penUp()
2140   PenDown = 0
2150 endfunc
2160 /*
2170 func penDown()
2180   PenDown = 1
2190 endfunc
```

のままで，10行〜にリスト1が読み込まれます。save@，load@はこのように複数のリストを結合するときに便利に使える命令です。

これでリスト1とリスト2の2つのプログラムが合体しましたから，リスト3を参照しながら全体を見直していけばいいだけです。リスト3のように途中から行番号を変えたい場合は，

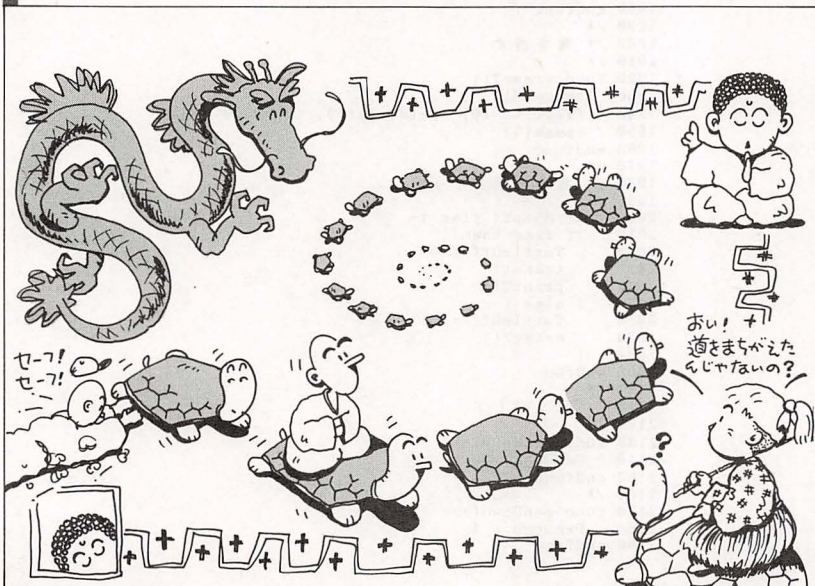
renum〔新しい行番号〕,〔古い行番号〕

とすればOKです。たとえば，400行以降の行番号を1000行から始めたいなら，「renum 1000,400」とします。110〜290行のプログラムは，三角形〜六角形

リスト4　コッホ曲線1

```
100 /*
110 init()
120 end
130 /*
140 func koch1( size )
150   forward( size )
160   left( 60 )
170   forward( size )
180   right( 120 )
190   forward( size )
200   left( 60 )
210   forward( size )
220 endfunc
900 /*
910 func goPoint(x,y,theta)
920   eraseT()
930   tx = x
940   ty = y
950   angle = theta mod 360
960   setA()
970   trans()
980   printT()
990 endfunc
```

リスト5　コッホ曲線2

```
230 /*
240 func koch2( size )
250   int mysize
260   mysize = size/3 + 0.5#
270   koch1( mysize )
280   left( 60 )
290   koch1( mysize )
300   right( 120 )
310   koch1( mysize )
320   left( 60 )
330   koch1( mysize )
340 endfunc
```

を順に表示します。ここでは亀を消していませんので，亀がチョコマカ動き回る様子を見て楽しむことができます。

135 dispT（0）

という1行を追加して，表示速度がどの程度違うか確かめてみるのもいいでしょう。110〜990行の間で亀の動きをプログラムして遊んでみてください。

不可思議な図形フラクタル

タートルグラフィックの応用プログラムをいくつか作ってみましょう。リスト4を見てください。これは先のリスト3に付け加えて（行番号が同じものは置き換えて）使うプログラムです。プログラムは110，120行で，亀の初期化を行い，終了するだけです。これは，リスト1と同じようにX-BASICのダイレクトモードで使うように作ってあります。リスト4を入力したら，まず「run」で実行し，「OK」と表示されたらダイレクトモードで，

goPoint（50，350，0）

として亀を（50，350）に移動させてください。この関数は亀を目的の位置に，目的の方向を向かせて移動させる関数です。亀を移動させたら次に，

koch1（150）

としてみてください。画面に山がひとつ現れましたね（写真1）。では次に，このプログラムにリスト5を追加してみてください。追加できたら再び「run」として実行したあと，今度は，

goPoint（50，350，0）

koch2（150）

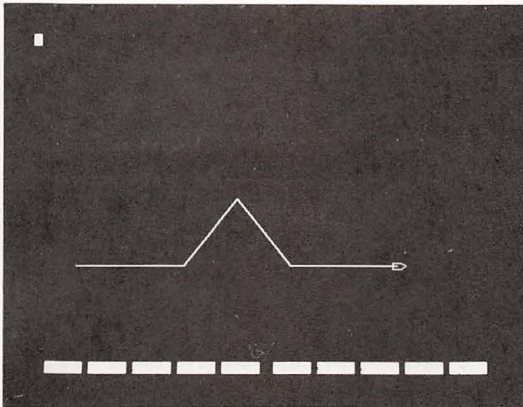
としてください。写真2のような図形が描かれました。よく見ると，写真1の山型の各辺を少し小さな同じ図形で置き換えた形になっています。もちろん，写真2の各辺をさらに置き換えることもできます。このように，自分と同じ形のもので自分の一部を置き換えた図形を「自己相似図形」といいます。一般にはフラクタル図形の名で知られているものです。

なぜこのような図形が描かれるのでしょうか。プログラムを見てみましょう。リスト4を見てください。山型を描いたのは140行から始まるkoch1という関数です。これは亀をsizeだけ進め，それから60°左を向いて進め，今度は120°右を向いて進め，最後に60°左を向いて進めるようになっています。これで山型がひとつできるわけですね。

では今度はリスト5を見てみましょう。koch2という関数は，mysizeという変数を用意し，引数として受け取ったsizeの1/3をセットします。そして，亀を進める代わりにkoch1を使って1/3の大きさの山型を描き，それから60°左を向いて1/3山型を描き……，

写真1

写真2

写真3

と続いていきます。koch1で直線を描いていたところが「山型を描く」という動作に変わったわけですから，写真2のような図形が描けたのです。山の中に山があるので，これを山山型と名付けましょう。この原理を使っていくと，koch3関数は「山型を描く」ところを「山山型を描く」に変えればうまくいきそうですね。実際それでうまくいきます。試してみてください。

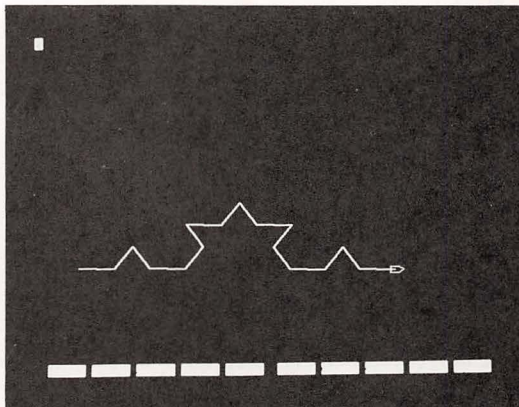
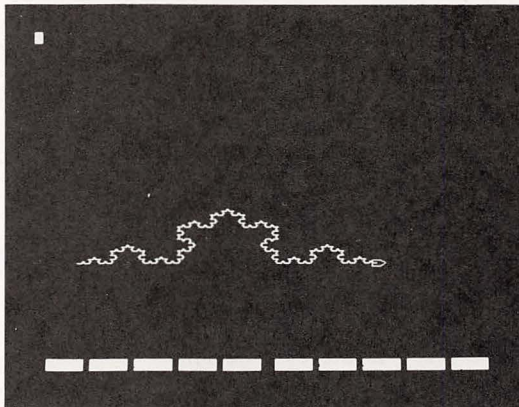
再帰を使って効率的に

この調子でkoch3，koch4，koch5と作っていけば，いくらでも複雑な山型を作ることができます。しかし，「山山山山山山山山山山型」を作るとなるとどうでしょう。とてもじゃないけど，そんなにたくさんのプログラムを打ち込む気にはなれません。

もう一度プログラムをよく見てみましょう。koch1とkoch2はどこが異なっているのでしょうか。山型に線を描くか，山型に山型を描くか，プログラムでいうなら，「forward(size)」を実行するか，「koch1(mysize)」を実行するかが違うだけです。その他の部分はまったく同じ形をしています。そこで，これを効率的にプログラムしてみたものがリスト6のkoch関数です。

このkoch関数は2つの引数をとります。ひとつはこれまでも使ってきた線の長さを表すsize，もうひとつは，何個の山を作ればいいかを表すtimesです。timesが1なら山型を，timesが2なら山山型を作ります。プログラムを追いながら動作を確認してみましょう。

まず180行でtimesが0かどうかを判定します。timesが0のときは，0個の山を作るわけですから，指定された長さの直線を引き，200行のreturnで終了します。timesが0でなければ，230行でtimes－1個の山を作り，それから60°左を向いてtimes－1個の山を作り……という作業を繰り返し行います。

例としてtimesが1の場合を考えてみましょう。timesは0ではありませんから，220行にきます。線の長さ÷3を計算したあと，230行でkoch(mysize, times－1)を実行しますが，times－1は0ですから，これはkoch(mysize, 0)となり，長さmysizeの直線を引くことに相当します。それから60°左を向き……と繰り返し処理は続きます。これはkoch1の動作と同じですね。

次にtimesが2の場合を考えてみましょう。timesは0ではありませんから，同じく220行にきます。mysizeを計算し，230行でkoch(mysize, 1)が実行されます。いま確認したように，koch(mysize, 1)は長さmysizeの山型の描くことになりますから，これは山山型を描くことになります。これは，koch2

リスト6　コッホ曲線（再帰）

```
100 /*
110 init()
120 goPoint(50,350,0)
130 koch(400, 4)
140 end
150 /*
160 func koch(size,times)
170   int mysize
180   if times = 0 then {
190     forward( size )
200     return()
210   }
220   mysize = size/3
230   koch( mysize, times-1 )
240   left( 60 )
250   koch( mysize, times-1 )
260   right( 120 )
270   koch( mysize, times-1 )
280   left( 60 )
290   koch( mysize, times-1 )
300 endfunc
```

と同じですね。

このように，プログラムの中で自分自身を呼び出して使うことを「再帰呼び出し」といいます。X-BASICでは，関数の中で宣言した変数は，その関数を実行している間だけ有効です。170行で宣言した変数mysizeは，関数の実行が始まると用意され，関数の実行が終わると破棄されます。何度でも関数が呼ばれるたびに，その関数の実行中だけ有効な新しいmysize変数が用意されます。

つまり，koch(size,2)を実行中に用意されたmysizeとは別にkoch(size,1)を実行中のmysizeが用意されるわけです。こうして，koch(size,2)で使っているmysize変数がkoch(mysize,1)の実行で変更されてしまうことがないようになっているのです。

このことは，リスト7のようなプログラムを作れば簡単に確認することができます。とりあえず入力して実行してみてください。図4のように表示されるはずです。40行から始まるsaiki関数は，関数の実行が始まると「begin(～)」のように現在のtimesを表示します。このときローカル変数colに5－timesをセットし，locate命令を使って何桁目から表示するかを決定しています[2]。

続いて100行でsaiki関数を再帰呼び出しし，呼び出しが終了するとcol桁目に今度は「end(～)」とtimesの値を表示して終了します。図4を見ると，begin(5)とend(5)は同じ桁から表示されていますね。これは再帰呼び出しをしても，colが変更されないからです。関数の引数は，呼び出し時に値が代入されるローカル変数として処理されますので，times変数の内容も再帰呼び出しの前後で保存されることになります。

2) csrlinは現在カーソルがあるy座標を保持している変数です。

リスト7
ローカル変数の表示例

```
 10 saiki(5)
 20 end
 30 /*
 40 func saiki( times )
 50   int col
 60   if times = 0 then return()
 70   col = 5-times
 80   locate col, csrlin
 90   print "begin(";times;")"
100   saiki( times-1 )
110   locate col, csrlin
120   print "end(";times;")"
130 endfunc
```

図4　リスト7の実行結果

```
RUN
begin( 5 )
 begin( 4 )
  begin( 3 )
   begin( 2 )
    begin( 1 )
    end( 1 )
   end( 2 )
  end( 3 )
 end( 4 )
end( 5 )
Ok
```

ドラゴン曲線の恐怖

再帰を利用して描ける図形の例として，次にドラゴン曲線を見てみましょう。ドラゴン曲線は，2つの関数から成り立っています。ひとつは直進し90°左に曲がって直進する，という動作を行います。もうひとつは，直進し90°右に曲がって直進するという動作です。ドラゴン曲線はこれら2つの関数を再帰的に使って描く曲線です。

その定義は，リスト8のようになります。左に曲がる関数をleftDragon，右に曲がる関数をrightDragonと命名しました。「ドラゴン曲線の恐怖」と題したのは，これら2つの関数がお互いに呼び出し合いながら再帰を行うからです。ちょっと見には実に複雑な動きを行いそうですが，例によってtimesが0のとき，1のとき，2のときと考えていくと，理解しやすいかと思います。頑張って挑戦してみてください。

ドラゴン曲線を描かせるには，コッホ曲線を描くリストを用意し，

delete 100－899

として不要な行を削除したあとリスト8を入力して実行すればOKです。90°左に曲がるか，90°右に曲がるかが違うだけの2つの関数によって，不思議な図形が描かれます。

図書館などでLOGOの参考書やコンピュータグラフィック関連の書籍を見れば，タートルグラフィックを利用する図形がこれ以外にもいろいろ掲載されていると思います。このタートルグラフィックシステムを利用して，自分で自由にプログラムしてみてください。

では，再見。

リスト8　ドラゴン曲線

```
100 /*
110 init()
120 leftDragon(5,10)
130 end
140 /*
150 func leftDragon(size,times)
160   if times = 0 then {
170     forward( size )
180     return()
190   }
200   leftDragon( size, times-1 )
210   left( 90 )
220   rightDragon( size, times-1 )
230 endfunc
240 /*
250 func rightDragon(size,times)
260   if times = 0 then {
270     forward( size )
280     return()
290   }
300   leftDragon( size, times-1 )
310   right( 90 )
320   rightDragon( size, times-1 )
330 endfunc
```

サブルーチンに汎用性を

Murata Toshiyuki 村田 敏幸

サブルーチンを使い回してプログラム作成の効率化を！ とはいっても，本当に汎用性のあるサブルーチンを作るにはそれなりの配慮が必要となります。ここではサブルーチンが使われる状況を細かくチェックしながら，さまざまなテクニックを招介しましょう。

今回は比較的大規模なプログラムを作る際に知っていると便利な（＝楽できる）技について話してみる。ユーザーライブラリの構築方法を中心に，その周辺技術までを紹介する。

ここでいうライブラリとは，汎用のルーチン群を指す。いつでも即使えるようなプログラムのパーツ集だ。当然，いざ大きなプログラムを作ろうと思った時点でライブラリを作り出すのでは手遅れで，日頃から汎用性の高いルーチン群を集めておかなければ意味がない。ライブラリは作ろうと思って作るのではなく，本来はいつの間にかできている性格のものだ。1本のプログラムを作るたびに，ほかのプログラムでも使えそうなパーツをピックアップしていくのがよい。もっとも，プログラムを書くときにある程度は意識していないと，なかなか使い回しの効くルーチンはできないというのもまた事実だ。

そういうわけだから，まずはその汎用性の高いルーチンの作り方あたりからつついてみる。

賢者(?)のサブルーチン

ある人がプログラムを作っていて，ふと「この処理は前に作ったプログラムにも出てきたぞ」と思ったとする。同じ処理を何度もプログラミングするのは非能率的なので，多少頭を巡らせて以前作ったルーチンを“流用”してやろうと考えた。ごく自然な発想だ。これを突き詰めていけばライブラリに行き着くわけだが，彼はそこまで深くは考えない。ディスクの中からしばらく前に作ったプログラムのソースを引っ張り出してきて，作成中のプログラムと一緒にエディタに読み込み，カット＆ペーストして必要な部分を抜き出せば，時間と労力の節約になるだろうぐらいに考えた。内心，自分はなんて冴えてるんだ，とか思う。

ところが，あいにくその“必要な処理”がメインルーチンの一部に埋め込まれていたとする。どこからどこまで抜き出せばよいのか探さなければならなくなった。作ってから1週間もたてば記憶もだいぶ薄れているだろうし，日頃からコメントをマメに付けていなかったりすると手がかりがまったくない。肝心な部分が見つかるかどうかは疑わしい。結局，彼は諦めた。

ここで第1の教訓が得られる。汎用性云々以前の問題で，当たり前すぎて恥ずかしいのだが，こういうことだ。

教訓1：あるルーチンをほかのプログラムでも使う可能性がある場合は，その部分を明確な処理単位であるサブルーチンにしておくのがよい。

もうひとつ無理やり教訓をひねり出すと，

教訓2：コメントはないよりあるほうがマシ。

となる。

さて，しばらくしてその彼がまた同じような事態に陥った。彼も少しは学習したので，今度は必要な処理はちゃんとサブルーチンにしてある。どんな働きをするサブルーチンかも簡単なコメントで示してあったので，目的のパーツはすぐに見つかった。彼はそのサブルーチンを抜き出して作成中のプログラムに持ってくる。

これで終わりだと思ったら甘かった。そのサブルーチンでは内部でいくつかのレジスタを使用しているのだが，いま作っているプログラムのメインルーチンでも同じレジスタを使っていたのだ。気づいたからよかったものの，もし見落としていたらとんでもないことになるところだ。彼はサブルーチンの先頭でmovemを使って内部で使用するレジスタをすべてスタックに待避し，rtsの直前で復帰するようにしてことなきを得た。今回は一応うまくいったくちだ。

ここでの教訓は，

教訓3：サブルーチンではレジスタを破壊しないよ

リスト1

```
 1: *       D0.Wを10進左詰めで表示するサブルーチン
 2:
 3:         .include        doscall.mac
 4: *
 5:         .text
 6:         .even
 7: *
 8: prtdec:
 9:         movem.l d0/a0,-(sp)     * {d0,a0をスタックに待避
10:
11:         andi.l  #$0000ffff,d0   *上位ワードをクリア
12:         lea.l   bufend,a0       *ポインタ初期化
13: prtdec0:
14:         divu.w  #10,d0          *d0.lを10で割る
15:         swap.w  d0              *上位ワードと下位ワードを交換
16:         addi.w  #'0',d0         *0～9 → '0'～'9'
17:         move.b  d0,-(a0)        *1桁格納
18:         clr.w   d0              *次の除算に備える
19:         swap.w  d0              *上位ワードと下位ワードを交換
20:         bne     prtdec0
21:
22:         move.l  a0,-(sp)        *変換した文字列を表示する
23:         DOS     _PRINT          *
24:         addq.l  #4,sp           *
25:
26:         movem.l (sp)+,d0/a0     * } d0,a0を復帰
27:         rts
28: *
29:         .data
30:         .even
31: *
32: buff:   .ds.b   5               *10進文字列格納領域
33: bufend: .dc.b   0               *文字列の終了コード
34:
35:         .end
```

うにするのが汎用性の第一歩。

ということにでもなるだろう。

ひと言つけ加えておくと，あるプログラム中のサブルーチンが汎用であるためには，ほかのルーチンから独立し，切り離されていることが必要条件になる。ほかのルーチンについて“知らなければ知らないほど”よい。ここでは無知は美徳である。メインルーチンでどのレジスタを使っているかなんて知らないほうが楽できる。サブルーチン内で使用するレジスタの値を待避するのはメインルーチンの顔を立てるためではない。サブルーチン側(とプログラマ)が楽するためである。

では，架空の名もない彼にはここで退場していただいて，次にこの連載で以前作ったサブルーチンの独立性・汎用性を試しに見てみよう。

サブルーチンの使用状況を考える

リスト1のprtdecは第5回で使った無符号数の10進表示サブルーチンだ。d0.wに表示したい数を入れて呼び出すとその数を左詰めで標準出力に書き出す。サブルーチン内部で使用するレジスタの値はスタックに保存してあるし，一応使い回しが効くような作りになっている。このサブルーチンを作ったときには“一応汎用性を狙ったので，結構がっちり作ってある”なんて偉そうなコメントがついていたりもする。

確かに，このサブルーチンが1つあれば，いつでも必要なときにd0.wの値を表示することができる。あたかも“d0.wの値を表示する”という命令があるかのように，だ。その目的においては便利このうえない。が，これをもって汎用を謳うのは嘘もいいところだ。このサブルーチンがそのまま使える場面はそう多くはない。

たとえば，表示したい数がd0.wではなくd1.wに入っていたとする。新たにd1.wの内容を表示するサブルーチンを作るのはいくらなんでも間抜けだから，次のようにd0.wにd1.wの値をいったん代入してからこのprtdecサブルーチンを呼び出すことになる。

```
move.w  d1,d0
bsr     prtdec
```

d0を介してパラメータをサブルーチンに引き渡すわけだ。

ところが，もしかするとd0.wは別の目的にすでに使われているかもしれない。その場合はd0.wの値を一時的に待避しておく必要があるので，

```
move.w  d0,-(sp)
move.w  d1,d0
bsr     prtdec
move.w  (sp)+,d0
```

のようにしなければならなくなる。

これはサブルーチンの呼び出し手順としては繁雑すぎるし，明らかにサブルーチンの独立性を損なっている。d0に値を入れてからサブルーチンを呼び出すという方法自体の問題だ。

また，値を標準出力ではなく標準エラー出力に書き出したいという場合や，値を左詰めではなく5桁の右詰めで表示したいという場合，あるいは値の符号を考慮して表示したい場合を考えてみよう。これらの要求に対してはprtdecサブルーチンはまったく役に立たず，“prtdecとほとんど同じだがごく一部が違うサブルーチン”をそれぞれ別々に用意しなければならない。

これは1つのサブルーチンとして切り出す処理単位の選び方がまずかったためだ。prtdecサブルーチンは処理単位としては大き過ぎたのだ。汎用を目指すのであれば，数値を文字列に変換する処理と，その文字列を表示する処理を切り離すべきだったといえる。

10進表示を行うサブルーチンだけがあるのと，数値→文字列変換ルーチンだけがあるのでは後者のほうがずっと幅広く使えるだろう。数値→文字列変換ルーチンを利用すれば10進表示ルーチンを作ることはできるが，逆はできないのだ[1]。

そこで，

教訓4：汎用性を目指すなら1個のサブルーチンにはただひとつの処理だけを行わせるようにしよう。

1) ネジ回しと並んで，大が小を兼ねないということを示す典型例である。

となる。

パラメータの渡し方

ここで，さっき問題になったサブルーチンへパラメータを渡す方法について，少し深く掘り下げてみよう。独立性という点だけではなく，速度効率・メモリ効率も比べてみたい。

リスト2にいくつかの例（a～g）を示す。どの例もサブルーチンsubにロングワードデータ1つとワードデータ1つを渡すものとしよう。便宜上，ロングワードのほうを第1パラメータ，ワードのほうを第2パラメータと呼ぶ。なお，サブルーチン中で使うレジスタの値を保存することは考えない。

●レジスタを使う

a)はさっきのprtdecのようにレジスタを介して値を渡す方法だ。もっとも直感的な方法といえる。レジスタに値をポンと入れてサブルーチンを呼び出すだけなので，処理ステップも短く，高速である。MPU68000にはレジスタがたくさんあるから，3つ4つのパラメータを渡すぐらいわけはない。

が，先に述べたように，パラメータの受け渡しに使用するレジスタとメインルーチンで使用するレジスタの競合をつねに気にしなければならず，サブルーチンの独立性は損なわれる。ある特定のプログラム中で頻繁に使われ（それゆえ呼び出しは高速に行われることが望ましい），かつ，ほかのプログラムに流用することを考えない（独立性・汎用性は捨ててかかる）といった限られた場合にのみ採用するとよいだろう。

●特定のワークエリアを使う

b）はレジスタの代わりにある特定のワークエリアに値を入れてからサブルーチンを呼び出すという方法だ。レジスタを使ったときのような“ぶつかり合い”を気にしなくてもすむし，まったく同じパラメータで再度サブルーチンを呼び出す場合は，前回ワークにセットした値をそのまま利用するという手抜き技が使える。ただ，パラメータを受け渡すという目的のためだけにワークエリアを用意するのはメモリの無駄遣いである。かといって，兼用しようなんて考えると今度はぶつかり合いを気にする必要が出てきてしまう。

●ポインタで渡す

c）はb）と似たような方法で，適当なメモリ領域にパラメータをセットし，その領域へのポインタをサブルーチンに渡すというものだ。パラメータが複数ある場合は連続したメモリに決まった順序でセットすることにすれば，ポインタは1つですむ。

パラメータの受け渡し用にメモリ領域が必要では

リスト2-a

```
 1: main:
 2:         move.l  #10000,d0       *第1引数セット
 3:         move.w  #$abcd,d1       *第2引数セット
 4:         bsr     sub             *サブルーチンコール
 5:
 6:          :
 7:
 8: sub:
 9:         *引数はもうd0.l,d1.wに入っている
10:
11:          :
12:
13:         rts
```

リスト2-b

```
 1: main:
 2:         move.l  #10000,par1     *第1引数セット
 3:         move.w  #$abcd,par2     *第2引数セット
 4:         bsr     sub             *サブルーチンコール
 5:
 6:          :
 7:
 8: sub:
 9:         move.l  par1,d0         *第1引数取り出し
10:         move.w  par2,d1         *第2引数取り出し
11:
12:          :
13:
14:         rts
15:
16: par1:   .ds.l   1               *第1引数受け渡し領域
17: par2:   .ds.w   1               *第2引数受け渡し領域
```

リスト2-c

```
 1: main:
 2:         move.l  #10000,par1     *第1引数セット
 3:         move.w  #$abcd,par2     *第2引数セット
 4:         lea.l   pars,a0         *引数受け渡し領域
 5:         bsr     sub             *サブルーチンコール
 6:
 7:          :
 8:
 9: sub:
10:         move.l  (a0)+,d0        *第1引数取り出し
11:         move.w  (a0),d1         *第2引数取り出し
12:
13:          :
14:
15:         rts
16:
17: par1:   .ds.l   1               *第1引数受け渡し領域
18: par2:   .ds.w   1               *第2引数受け渡し領域
19:                                 *本当ならこんな固定領域は不用
```

――サブルーチンからの戻り値――

本文ではサブルーチンへパラメータを渡すことばかり解説したが，逆にサブルーチンからメインルーチンへ値を返す方法も検討しておく必要があるだろう。結論からいうと，サブルーチンへパラメータを渡す方法のほとんどを，戻り値を返す方法として利用することができる。ここでも独立性を追求するならスタックに値を積んで返す方法がよいのだが，これには結構余分なスタック操作を必要とする（考えてみるように）。

そのためかどうかは知らないが，現実にはメモリ効率や実行速度を優先し，戻り値はレジスタで返す場合が多い。この場合，サブルーチンの独立性が下がることになるが，つねに同じレジスタで値を返すといった自分なりの規則を作れば，それほど酷いことにはならない。

リスト2-d

```
 1: main:
 2:         move.l  #10000,par1     *第1引数セット
 3:         move.w  #$abcd,par2     *第2引数セット
 4:         bsr     sub             *サブルーチンコール
 5: par1:   .ds.l   1               *第1引数受け渡し領域
 6: par2:   .ds.w   1               *第2引数受け渡し領域
 7: retn:
 8:          :
 9:
10: sub:
11:         movea.l (sp)+,a0        *引数へのポインタ
12:         move.l  (a0)+,d0        *第1引数取り出し
13:         move.w  (a0)+,d1        *第2引数取り出し
14:         move.l  a0,-(sp)        *リターンアドレスセット
15:
16:          :
17:
18:         rts
```

リスト2-e

```
 1: main:
 2:         move.l  #10000,par1+2   *第1引数セット
 3:         move.w  #$abcd,par2+2   *第2引数セット
 4:         bsr     sub             *サブルーチンコール
 5:
 6:          :
 7:
 8: sub:
 9: par1:   move.l  #$aaaaaaaa,d0   *第1引数取り出し
10: par2:   move.w  #$bbbb,d1       *第2引数取り出し
11:
12:          :
13:
14:         rts
```

あるが，パラメータの所在はポインタで示されるので，空いているメモリならどこでも使える。アドレスレジスタをパラメータ受け渡し専用に1個つぶす覚悟があれば，効率も悪くはないだろう。しかし，最終的にアドレスレジスタでポインタを渡す限り，独立性という意味では a) と大差がない。

●プログラムの一部を使う

d) はパラメータをプログラムの途中に埋め込むという方法だ。サブルーチンを呼び出すbsrの直後に必要なだけのメモリを用意し，ここにパラメータをセットしておく。ちょっと見にはこのパラメータの部分も命令と見なして実行してしまいそうに見えるが，そこはそれ，サブルーチン側でつじつまを合わせ，ちゃんとスキップするようにできている。

サブルーチンsubを呼び出した時点で，スタックトップにはサブルーチンからのリターンアドレスが積まれている。リスト2-dの場合はラベルpar1で示されるアドレスがスタックトップにあるわけだ。これを適当なアドレスレジスタに取り出すと，そのアドレスレジスタはちょうどc) のパターン同様パラメータ群を指すポインタとなる。このポインタをポストインクリメントし，順にパラメータを取り出していくと，全部のパラメータを取り出した時点でこのポインタはパラメータ群の直後（リスト2-dではラベルretnで示されるアドレス)を指す。このアドレスは“サブルーチンから戻って最初に実行すべき命令”のアドレスになっており，スタックに積み直したうえでrtsすれば望みの位置から処理を再開できることになる。

この方法はパズルとしては面白いが，プログラムサイズがパラメータ受け渡し領域の分だけ間延びする，パラメータを取り出してリターンアドレスを再設定するまでの間スタックがあまり自由に使えない（やってできないことはないが）などの欠点もあり，68000ではあまり使われることはない。

●プログラム自体を書き換える

e) はもっと強引な方法だ。パラメータ受け渡し領域はプログラム中に埋め込まれるが，余計なメモリ領域を必要としない。どうやるかというと，直接マシンコードのオペランド部分を書き換えるのだ。いわゆる“自己書き換え”に類するテクニックで，使用するにあたってはアセンブリ言語レベルだけではなく，マシンコードレベルの知識が必要だ。つまり，アセンブリ言語で書いたプログラムがどのようなコードに変換されるかを知っていなければならない。

リスト2-eでは2行と9行，3行と10行が対になっている。2，3行がパラメータを渡す部分で，9，10行がそのパラメータを受け取る部分だ。2行ではpar1+2で示されるアドレスにロングワードでパラメータをセットしている。+2がどこから出てきたのかは，9行の，

move.l #$aaaaaaaa, d0

が，アセンブルすることによって，

2030 aaaa aaaa

という3ワードのコードに変換されることと関係がある。最初の1ワードは，

move.l #～, d0

を意味し，残りの2ワードでロングワードの即値aaaaaaaaHを表している。2行でこの後半2ワードを書き換えることで，任意の値をd0.lに代入する命令に組み換えることができるわけだ。

この方法の利点としては，場合によっては速度・メモリの効率がよい場合があるとか，パラメータのセットと実際のサブルーチン呼び出しが必ずしも連続していなくてもよいとかがある。が，マシンコードレベルの知識が必要という最大の欠点のために，低機能なマイクロプロセッサならともかく68000ではこんなことをするのはバカげている。68000にはもっとエレガントな方法が似つかわしい。

●スタックに積む

f) がその68000らしいエレガントな方法というやつだ。パラメータはスタックに積んでサブルーチンに渡す[2)]。DOSコールの呼び出し手順でお馴染みの方法だろう。エレガントというよりは重厚さを感じさせる。はっきりいってレジスタで直接渡すような方法と比べると速度的にはいささか不利ではある。

2) スタックを使用するのだから正確にはspを間接的に利用することになるのだが，少なくとも表には出ないし，意識する必要もあまりない。

半面，レジスタやワークエリアをいっさい使用しないので，汎用性を狙うにはベストの選択となる。

細かく見てみよう。サブルーチンsubが呼び出された時点でのスタックの状態は図1のようになっている。スタックトップには，例によってサブルーチンからのリターンアドレスが積まれている。その次に，積んだ順序の逆順でパラメータが並ぶ。この点を考慮してリスト2-fではサブルーチンにパラメータを渡す順序がほかの例とは見掛け上反転している。

一般には，スタックからデータを取り出すには，そのデータがスタックトップになければならない。第1パラメータを取り出すためにはリターンアドレスが邪魔だ。真正直にやると，一度スタックトップに積まれたリターンアドレスを適当なレジスタなりメモリなりに取り出してから，パラメータを順次ポップすることになるだろう。

しかし，68000のspはそんじょそこらのマイクロプロセッサのスタックポインタとはわけが違う。68000のspはa7レジスタの別名であり，アドレスレジスタのうちの1本だ。当然，アドレスレジスタに関するアドレッシングモードもすべて使える。ここで，“ディスプレイスメント付きアドレスレジスタ間接アドレッシング”を思い出してもらえれば話は早い。このアドレッシングモードは“アドレスレジスタに変位（ディスプレイスメント）を加えたアドレス”を指定するものだった。アドレスレジスタが指しているアドレスそのものではなく，その前後に少しずれた位置を指定するのに利用する[3]。

そこで，もう一度図1を見てもらおう。第1パラメータは現在のspに4を，第2パラメータは8を足したアドレスに置かれている。それぞれのアドレスは，“ディスプレイスメント付き～”を使えば“4 (sp)”，“8 (sp)”のように表現される。あとは10, 11行のようにmove命令一発でレジスタにパラメータを取り出すことができる。

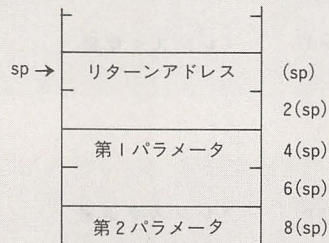
なお，この場合サブルーチンから戻った時点で，スタックにパラメータが積まれたままになっていることに注意したい。DOSコール呼び出しのときと同様，スタックポインタの補正が必要になる。

●スタックにポインタを積む

おまけとして，g）はc）とf）を合わせたようなものだ。パラメータは適当なメモリ領域にセットし，そのポインタをスタックに積むことでサブルーチンへと渡す。サブルーチンでのパラメータ受け取り手順は，ポインタを取り出すところまでがf）と同じで，さらに実際のパラメータを取り出すにはc）と同様の手順を踏む。この方法は大量のパラメータをスタックを介してサブルーチンに引き渡す際のヒントになっている。

リスト2-f

```
 1: main:
 2:           move.w   #$abcd,-(sp)     *第2引数セット
 3:           move.l   #10000,-(sp)     *第1引数セット
 4:           bsr      sub              *サブルーチンコール
 5:           addq.l   #6,sp            *sp補正
 6:
 7:            :
 8:
 9: sub:
10:           move.l   4(sp),d0         *第1引数取り出し
11:           move.w   8(sp),d1         *第2引数取り出し
12:
13:            :
14:
15:           rts
```

図1　スタックを使ったパラメータの受け渡し

リスト2-g

```
 1: main:
 2:           move.l   #10000,par1      *第1引数セット
 3:           move.w   #$abcd,par2      *第2引数セット
 4:           pea.l    par1             *引数受け渡し領域
 5:           bsr      sub              *サブルーチンコール
 6:           addq.l   #4,sp            *sp補正
 7:
 8:            :
 9:
10: sub:
11:           movea.l  4(sp),a0         *引数へのポインタ
12:           move.l   (a0)+,d0         *第1引数取り出し
13:           move.w   (a0),d1          *第2引数取り出し
14:
15:            :
16:
17:           rts
18:
19: par1:     .ds.l    1                *第1引数受け渡し領域
20: par2:     .ds.w    1                *第2引数受け渡し領域
```

●

以上，サブルーチンへパラメータを渡す方法を何通りか比較検討してみた。結論としては，速度重視ならa）のレジスタを使う方法，サブルーチンの汎用性重視ならf）のスタックを使う方法ということになると思う。そのバリエーションとしてc）g）のポインタを使う方法も併用されるだろう。

今回はサブルーチンの汎用性を追求する主旨だから，一応スタックにパラメータを積むパターンを採用し，以下，さらに詳しく見ていくことにする。

3）補足しておくと，“ディスプレイスメント付き～”で使用可能な変位は，符号付き16ビット数で表せる範囲（－32768～＋32767）だ。変位をアドレスレジスタに加算して実効アドレスを求める際には，変位は32ビット長に符号拡張される。

スタックに積む場合の詳細

残念なことにリスト2-fのままでは，プログラミング上の問題が残っている。試しにリスト2-fにサブルーチン内で使用するレジスタの待避・復帰処理を付

け加えてみてほしい。仮にd0～d7のデータレジスタすべてを保存するものとしよう。

リスト3-aのようにmovemを2カ所に挟み込めばいいように思えるところだが，そうではない。正しくはリスト3-bのようにパラメータ取り出し部分にも手を加える必要がある。待避したレジスタの分だけスタックポインタがずれてしまうからだ。

リスト3-a

```
 1: sub:
 2:         movem.l d0-d7,-(sp)      *レジスタ退避
 3:
 4:         move.l  4(sp),d0         *第1引数取り出し？
 5:         move.w  8(sp),d1         *第2引数取り出し？
 6:
 7:          :
 8:
 9:         movem.l (sp)+,d0-d7      *レジスタ復帰
10:         rts
```

リスト3-b

```
 1: sub:
 2:         movem.l d0-d7,-(sp)      *レジスタ退避
 3:
 4:         move.l  4+4*8(sp),d0     *第1引数取り出し
 5:         move.w  8+4*8(sp),d1     *第2引数取り出し
 6:
 7:          :
 8:
 9:         movem.l (sp)+,d0-d7      *レジスタ復帰
10:         rts
```

リスト3-c

```
 1: sub:
 2:         movem.l d0-d7,-(sp)      *レジスタ退避
 3:
 4:         move.l  4+4*8(sp),d0     *第1引数取り出し
 5:         move.w  8+4*8(sp),d1     *第2引数取り出し
 6:
 7:                  :
 8:
 9:         move.l  d2,-(sp)         *レジスタ一時退避
10:         move.l  4+4*8+4(sp),d2   *第1引数再取り出し
11:
12:                  :
13:
14:         movem.l (sp)+,d0-d7      *レジスタ復帰
15:         rts
```

リスト4-a

```
 1: sub:
 2:         movea.l sp,a6            *スタックフレーム生成
 3:         movem.l d0-d7,-(sp)      *レジスタ退避
 4:
 5:         move.l  4(a6),d0         *第1引数取り出し
 6:         move.w  8(a6),d1         *第2引数取り出し
 7:
 8:          :
 9:
10:         movem.l (sp)+,d0-d7      *レジスタ復帰
11:         rts
```

リスト4-b

```
 1: sub:
 2:         move.l  a6,-(sp)
 3:         movea.l sp,a6            *スタックフレーム生成
 4:         movem.l d0-d7,-(sp)      *レジスタ退避
 5:
 6:         move.l  8(a6),d0         *第1引数取り出し
 7:         move.w  12(a6),d1        *第2引数取り出し
 8:
 9:          :
10:
11:         movem.l (sp)+,d0-d7      *レジスタ復帰
12:         move.l  a6,-(sp)
13:         rts
```

このことはサブルーチンにわずかの手を入れることさえ難しくする。たとえば，サブルーチンに処理を付け加えた結果，新たにa0レジスタも保存しなければならなくなったとしよう。movemのレジスタリストにa0を付け加えるときに，パラメータ取り出し部分も変更する必要があるのを忘れる可能性は高い。こういうミスをあとから見つけることはかなり難しいだろう。

また，リスト3-cのようにスタックに値をいくつか積んでから改めてパラメータを取り出すような形になっていたりすると，さらに複雑さが増す。ある瞬間にスタックにどれだけのデータが積まれているかをいちいち計算しないと変位が求められないのだ。

どうしてこんなことになったかといえば，基準として選んだspが頻繁に変更されてしまうからだ。もちろん，スタックを極力使わないようにすればどうにかなるが，スタックを使わずにマシン語プログラムを書くなんて拷問に等しい。

そこで，スタックに代わる別の基準を作ることを考える。たとえば，リスト4-aのようにサブルーチンが呼び出された直後のspを適当なアドレスレジスタに入れてしまうというのはどうだろう。リスト4-aではa6を使ってみた。パラメータを取り出す部分はspがa6に変わっただけだし，以下，spがどんなに変動しても，a6さえ固定しておけば，パラメータはいつでも“同じ場所”にあるから安心だ。

これはスタック上に枠組みを作ってやるようなもので，その意味で確保された領域のことをスタックフレームと呼ぶ。また，スタックフレームのベースアドレス（基準となるアドレス）を保持するレジスタはフレームポインタと呼ばれたりもする。

この方法を採ることでサブルーチンは作りやすく，修正しやすくなる。また，ディスプレイスメントをラベル定義しておけばプログラムの読みやすさも上がるというオマケもついてくる。弊害でアドレスレジスタが1本使えなくなるが，太っ腹の68000にとっては制限にすら感じられない。

実際にはリスト4-aのままではa6が破壊されてしまうから，少し細工してリスト4-bのようにしたほうがよい。あらかじめa6をスタックに待避してからspの値を代入するわけだ。この場合，a6をスタックに積んだ分“a6に代入した時点でのsp”は4バイトずれていることになる。それに伴ってスタック上のパラメータの位置も変わってくるが，サブルーチンの中では一定だから問題ではない。

サブルーチンの最初と最後に余分な命令がいくつも並ぶのは気持ちのよいものではない。わずかとはいえプログラムが長くなればそれだけ間違いの入り

込む余地も大きくなる。だからというわけではないのだろうが，68000にはスタックフレームの生成・解放を行う専用命令がある。もともとはC言語などの高級言語をサポートしやすいように用意されたもののようだ。スタックフレームの生成はlink，解放はunlk（unlinkを詰めてある）という命令で行う。これを利用するとリスト4-bはリスト4-cのようにすっきりしてくる。

linkは次のようにして使う。

link　　アドレスレジスタ，#ローカルエリアサイズ

第2オペランドに即値がくるという68000の命令としては例外的な語順になっていることに注意。

この命令はローカルエリアサイズに0を指定するとリスト4-bの2～3行と同じような働きをする。具体的には次のような動作だ。

1）指定されたアドレスレジスタをスタックに待避する。
2）1）のあと，spの値を指定のアドレスレジスタに代入する（すでにアドレスレジスタをスタックに積んだ分だけspは更新されている）。

本当はこののち，

3）spに第2オペランドのローカルエリアサイズを加える。

という動作があるのだが，ローカルエリアサイズが0であれば同じことだ。なお，ローカルエリアに関しては107ページのコラムを参照のこと。

対するunlkは，

unlk　　アドレスレジスタ

のようにして使う。アドレスレジスタは通常linkで指定したのと同じものを指定する。単にlinkで生成したスタックフレームを解放する命令だと思っていれば間違いない。一応，具体的な動作を示すと，

1）指定されたアドレスレジスタの値をspに代入する。
2）スタックトップのロングワードデータを指定のアドレスレジスタにポップする。

という動きだ。

link，unlkで指定するアドレスレジスタはa7でさえなければなんでもよいのだが，慣例としてa6が使われる。以後はこれにならうことにする。

prtdecルーチンを改良する

ここで再びprtdecサブルーチンに立ち戻る。ここまでの話ではっきりした問題点を整理し，より汎用性のあるルーチンに改造してみた。結果はリスト5のようになった。深い意味はないがサブルーチン名はputdecに変更されている。

リスト4-c

```
 1: sub:
 2:         link     a6,#0              *スタックフレーム生成
 3:         movem.l d0-d7,-(sp)         *レジスタ退避
 4:
 5:         move.l  8(a6),d0            *第1引数取り出し
 6:         move.w  12(a6),d1           *第2引数取り出し
 7:
 8:          :
 9:
10:         movem.l (sp)+,d0-d7         *レジスタ復帰
11:         unlk    a6
12:         rts
```

リスト5

```
 1:         .include        const.h
 2:         .include        doscall.mac
 3: *
 4:         .text
 5:         .even
 6: *
 7: *       16ビット無符号整数を10進左詰めで表示する
 8: *
 9: *               +----------------+
10: *       (a6)    |      旧a6      |     <-sp
11: *               +................+
12: *               |                |
13: *               +----------------+
14: *       4(a6)   |リターンアドレス|
15: *               +................+
16: *               |                |
17: *               +----------------+
18: *       8(a6)   |   表示する値   |
19: *               +----------------+
20: *
21: value   =       8
22: *
23: putdec:
24:         link    a6,#0              *スタックフレーム生成
25:
26:         move.w  #STDOUT,-(sp)
27:         move.w  value(a6),-(sp)
28:         bsr     fputdec
29:         addq.l  #4,sp
30:
31:         unlk    a6
32:         rts
33:
34: *
35: *       16ビット無符号整数を10進左詰めでファイルに出力する
36: *       d0.lにDOSコールfputsの終了コードを持って戻る
37: *
38: *               +----------------+
39: *       -6(a6)  |                |     <-sp
40: *               +................+
41: *               |                |
42: *               +................+
43: *               |                |
44: *               +----------------+
45: *       (a6)    |      旧a6      |
46: *               +................+
47: *               |                |
48: *               +----------------+
49: *       4(a6)   |リターンアドレス|
50: *               +................+
51: *               |                |
52: *               +----------------+
53: *       8(a6)   |   表示する値   |
54: *               +----------------+
55: *       10(a6)  |ファイルハンドル|
56: *               +----------------+
57: *
58: buffsiz =       6
59: temp    =       -buffsiz
60: value   =       8
61: fno     =       10
62: *
63: fputdec:
64:         link    a6,#-buffsiz       *スタックフレーム生成
65:                                    *+ローカルエリア確保
66:
67:         pea.l   temp(a6)           *数値→文字列変換
68:         move.w  value(a6),-(sp)    *
69:         bsr     utoa               *
70:         addq.l  #6,sp              *
71:
72:         move.w  fno(a6),-(sp)      *変換後の文字列を出力
73:         pea.l   temp(a6)           *
```

```
 74:          DOS      _FPUTS             *
 75:          addq.l   #6,sp              *
 76:
 77:          unlk     a6                 *スタックフレーム解放
 78:          rts
 79:
 80: *
 81: *        16ビット無符号整数を文字列に変換する
 82: *
 83: *                 +---------------+
 84: *        (a6)     |     旧a6      |       <-sp
 85: *                 +...............+
 86: *                 |               |
 87: *                 +---------------+
 88: *        4(a6)    |リターンアドレス|
 89: *                 +...............+
 90: *                 |               |
 91: *                 +---------------+
 92: *        8(a6)    |      値       |
 93: *                 +---------------+
 94: *        10(a6)   | 文字列格納領域 |
 95: *                 +...............+
 96: *                 |  へのポインタ |
 97: *                 +---------------+
 98: *
 99: value    =        8
100: buff     =        10
101: *
102: utoa:
103:          link     a6,#0              *スタックフレーム生成
104:          movem.l  d0/a0-a1,-(sp)     *{レジスタ待避
105:
106:          moveq.l  #0,d0              *d0.l=表示する数
107:          move.w   value(a6),d0       *
108:          movea.l  buff(a6),a1        *a1=文字列格納領域先頭
109:          lea.l    5(a1),a0           *a0=文字列格納領域最終
110:          clr.b    (a0)               *終端コード書き込み
111:
112: utoa0:   divu.w   #10,d0             *d0.lを10で割る
113:          swap.w   d0                 *上位ワードと下位ワードを交換
114:          addi.w   #'0',d0            *0～9 → '0'～'9'
115:          move.b   d0,-(a0)           *1桁格納
116:          clr.w    d0                 *次の除算に備える
117:          swap.w   d0                 *上位ワードと下位ワードを交換
118:          bne      utoa0
119:
120: utoa1:   move.b   (a0)+,(a1)+        *左詰めにする
121:          bne      utoa1              *
122:
123:          movem.l  (sp)+,d0/a0-a1     *}レジスタ復帰
124:          unlk     a6                 *スタックフレーム解放
125:          rts
126:
127:          .end
```

新しいputdecルーチンの機能自体は基本的にはこれまでと変わらない。16ビット無符号数を10進左詰めで標準出力に書き出す。ただ，以前はd0.wで渡していたパラメータはスタックを介して渡すように修正されている。

リストを見ると，いままでは単独のサブルーチンであったものが3つのサブルーチンに分離されており，見掛け上の複雑さは増したように見えるかもしれない。こうなってしまったのは処理単位を細かく一般化して，処理の階層関係をはっきりさせたためだ。数値を標準出力に書き出すという処理は，数値を指定のファイルハンドルに書き出す処理というもっと一般的なケースの一部だ。そして，この処理は数値を文字列に変換する処理と，その文字列をファイルハンドルに書き出す処理に分離することができる。

23行以下のputdecは表示する値を取り出したら，「この値をここに出力してね」と言いながら，値と標準出力のファイルハンドルをパラメータとしてfputdecというサブルーチンを呼び出す[4]。彼の仕事はこれで終わりだ。無知とはいっても，“自分の仕事を誰に頼めばうまくやってくれるか”だけは知っていたわけだ。

fputdecは数値を文字列に変換する処理をutoa[5]というサブルーチンにさせてから，変換後の文字列を指定のファイルハンドルに出力している。数値を変換してできる文字列を格納するために一時的な作業領域が6バイト（最大5桁＋終端コード）必要だから，秘密主義に従ってlinkでローカルエリアを用意している。このときのスタックフレームの様子はリスト中の注釈を参考にしてもらいたい。

数値→文字列変換処理を行うutoaは，変換すべきワードデータと変換後の文字列を格納する領域へのアドレスをパラメータとして必要とする。変換処理のアルゴリズムは以前のputdecに埋め込まれていたものと同じだが，わけあって多少複雑さを増し，文字列へ変換してからズリズリと左詰めにする処理が入っている。

ソースは分割して利用しよう

ようやく汎用性の高いサブルーチンの作り方が固まった。しかしいまのところ，せっかく作った汎用ルーチンはソースレベルでプログラムに組み込んでアセンブルしなければ使えない。putdecのようにひとつの機能が複数のサブルーチンにばらされているような場合は，それぞれのサブルーチンを一緒に移動させる必要がある。これではあんまりなので，実際に使うときの手間暇を減らす工夫をしてみたい。

ここで分割アセンブルの考え方を導入する。1本のプログラムを複数のソースに分割しておき，個別にアセンブルしたうえで，最終的にリンク時に結合するという考え方だ。長いプログラムを十分見通しが効く大きさのソースに分けて開発したり，数人でプログラムを作るようなときにも便利だ。修正が入らない限り，各ソースはただ一度アセンブルしておけばよいわけで，開発時間の短縮にもつながる。

この応用で，汎用のサブルーチンは個別にアセンブルしておくことにし，アセンブル前ではなくリンクするときに組み込むことを考える。ソースを切り張りする手間と，もうできているサブルーチンを何度もアセンブルし直す時間をなくそうという魂胆だ。

ところが，通常，識別子（ラベル）を使うためには，同一のソース内のどこかで宣言しておかなければならないことになっている。経験済みと思うが，宣言されていないラベルを使おうとすると，アセンブラは困って停止してしまうのだ。

そこで，アセンブラに“このラベルはこのソース

4) スタックフレーム上のパラメータの位置は21行でラベル定義している。“=”はsetの省略形で，equのようにラベルを定義するのに使う疑似命令だ。equで定義したラベルは再定義することができないが，setで定義した場合は何度でも定義し直すことができる。プログラムの一部でのみ使用するようなラベルを定義するのに利用する。

5) utoaは，Unsigned integer TO Array of characterぐらいのつもりで，無符号整数を文字の配列状データ（ようするに文字列）へ変換するという意味での命名だ。Cでよく使われるハナモゲラである。

にはないよ”ということを教えるための疑似命令.xrefと，“このラベルは別のソースでも使うよ”という疑似命令.xdefを使う。

.xref,.xdefは，

　　.xref　ラベル名

　　.xdef　ラベル名

のようにして使用する。それぞれ“外部参照名”の定義，“外部定義名”の宣言を行う疑似命令だ[6)]。ラベル名は“,”で区切って複数並べることができる。

使い方は単純で，ソース外のラベルを参照するとき（サブルーチン呼び出しを含む）にはそのラベル名を.xrefで宣言し，参照される側では同名のラベルを.xdefで宣言すればよい。

.xrefと.xdefの使い分けが面倒であれば，.globl疑似命令を使ってもかまわない。これは大域的なシンボルの宣言を行う命令で，ちょうど.xrefと.xdefの機能を合わせたものだと思えばよいだろう。外部参照と外部定義のどちらも.globlで宣言してしまうことができる。

リスト6に.xrefと.xdefの使用例を示す。a)をリスト5の頭に付けてアセンブルし，b)は単体でアセンブルする。できた2つのオブジェクトファイルをPUTDEC.O,MAIN.Oとすると，

　　A>LK MAIN PUTDEC

によってリンクすれば，2つのオブジェクトが組み合わされ，MAIN.Xが生成される。

ここで，分割アセンブルに関連し，コラム『セクション』でいままで枕詞的に使ってきた.textなどの疑似命令の意味についてまとめておく。

リスト6-a

```
 1:          .xdef   putdec
 2:          .xdef   fputdec
 3:          .xdef   utoa
```

リスト6-b

```
 1:          .xref    putdec              *外部参照定義
 2: *
 3:          .include         doscall.mac
 4: *
 5:          .text
 6:          .even
 7: *
 8: ent:
 9:          lea.l    mysp,sp             *spの初期化
10:          move.w   #12345,-(sp)        *パラメータを積み
11:          bsr      putdec              *サブルーチンを呼ぶ
12:          addq.l   #2,sp               *スタック補正
13:
14:          DOS      _EXIT
15: *
16:          .stack
17:          .even
18: *
19: mystack:                              *スタック領域
20:          .ds.l    256                 *ローカルエリアの分も
21: mysp:                                 *計算に入れる！
22:          .end
```

ライブラリの作成

もう一歩突っ込もう。いまここに，リスト5+6-aをアセンブルしてできたオブジェクトファイルがある。この中には3つのサブルーチンが含まれている。それぞれは独立しても使用できる（実際には下位のサブルーチンも必要だが）ような作りになっているから，utoaだけを使いたいという状況も考えられる。ところが，リンクはファイル単位で行われるので，utoaだけをリンクしようとしても残り2つのサブルーチンがくっついてくる。この余分なサブルーチンは実行ファイルのサイズを大きくすることのみにしか貢献しない[7)]。

こういうことが起こらないようにするためには，1つのオブジェクトファイルにはサブルーチンが1つしか存在しないよう分割する必要がある。リスト5+6-aの場合だと，サブルーチンごとにputdec.s,fputdec.s,utoa.sの3つに分割し，個別にアセンブルしておくことになるだろう（外部定義・参照に気をつけて実際に分割してみよう）。

6）逆に，外部定義しないほかのすべてのラベルはそのソース内でのみ有効になる。ほかのソースファイルに同名のラベルがあってもかまわないということだ。

7）プログラムに含まれてはいるが使われることのない部分を意味する“Dead Code”という言葉があるらしい。

セクション

これまで詳しい説明もなく使ってきた.text,.data,.bss,.stackはセクションを指定する疑似命令だ。それぞれ，テキストセクション，データセクション，ブロックトレースセクション，スタックセクションの始まりを表す。セクションの選択は次にこれらの疑似命令が現れるまで有効だ。

テキストセクションには実際に実行するプログラム，データセクションには“初期値付き”データ部（.dc），ブロックトレースセクションには“初期値なし”データ部（.ds），スタックセクションにはスタック領域を置く。これらは必ずしも守らなければならないというものではない。べつにテキストセクションにデータを置いたりしてもかまわないのだ。実際のところHuman68kではセクションはあまり深い意味を持たないといってもいいだろう。慣例というか，メーカーの顔を立てるぐらいのつもりでいればよい。

ただ，以前も触れたように，ブロックトレースセクションとスタックセクションに置かれた.ds命令は実行ファイルの大きさに影響を与えないという利点がある。テキストセクションに.dsで大きな領域を確保すると，実行ファイルにはその数だけの0が埋め込まれるが，ブロックトレースセクション，スタックセクションでは“何バイト確保されたか”という情報のみが残り，メモリに読み込まれた時点で展開される。

また，ひとつのプログラムの中に複数の同一セクションがある場合は，最終的にはそれらはみんなひとつにまとめられることになっている。だからどうなんだと言われても困るが。

なお，いつも.textたちとセットで使ってきた.even疑似命令については別コラム参照のこと。

ところが，こうしてしまうとputdecを使いたいときに，残りの2つも忘れずにリンクしてやらなければならなくなる。こんな感じだ。

A>LK MAIN PUTDEC FPUTDEC UTOA

このうちひとつをリンクし忘れたりすると，LK.Xはのろのろとしばらく待たせたあげく，“知らないラベルがある”と怒り出すわけだ。

ここで，登場するのがライブラリファイルだ。これはいくつものオブジェクトファイルをひとつにまとめたものである。上記の3つのオブジェクトファイルをまとめたライブラリファイル（仮に“USERLIB.A”）を作っておけば，リンク作業は，

A>LK MAIN USERLIB.A

だけですむ[8)]。LK.Xはこのライブラリファイルの中からMAIN.Oで使われているものだけを抜きとり，必要ならさらに下位のサブルーチンも探してリンクしてくれる。ライブラリの中に今回はまったく使わないほかのオブジェクトがあっても無駄は生じない。LK.Xはいらないオブジェクトはリンクの対象から外すからだ。

あとは具体的なライブラリファイルの作り方を示せば今月の話は終わりだ。

一般にはライブラリはライブラリアンと呼ばれるプログラムによって作成される。ところが，いまのところHuman68kにはまともなライブラリアンが用意されておらず，アーカイバで代用している。アーカイバは単に複数のファイルを詰めてひとつのファイルにまとめるものであり，リンカの都合を考えてくれない。リンカは必要なものを探すのに頭から順に見ていくしかないので，リンクに余計な時間がかかる。まともなライブラリアンがあれば，もう少しリンク作業が楽になるような形式のライブラリを作ってくれ，リンク時間も短くなるはずなのだ[9)]。

というわけで，ここではとりあえずアーカイバAR.X[10)]によるライブラリファイル（もどき）の作り方を簡単に紹介しておく。将来ライブラリアンが出てきたとしても使い方はそう変わらないはずだ。

基本的には，

A>AR ライブラリファイル名 オブジェクトファイル名 …

というようにライブラリファイル名の後ろにずらずらオブジェクトファイル名を並べればよい。これにより，指定されたライブラリファイルが生成される。もしすでに同名のライブラリファイルがあった場合は，オブジェクトの追加・置き換えが行われる。

あと，AR.Xにはスイッチによってライブラリから一部を抽出したり，削除したり，リストを取ったりできるといったひと通りの機能が揃っているから，必要に応じて『アセンブラマニュアル』（か，AR.Xのヘルプ）で調べてもらいたい。

●

今回は“あとで楽するためにいま苦労する”法としてのユーザーライブラリ作成技術を紹介し，そのための汎用サブルーチンの作り方をごりごりと説明してきた。汎用性・独立性にこだわると，どうしても速度・メモリ使用の効率は下がってしまうが，大きなプログラムも比較的楽に作れるのは大きい。

かといって，あまりに開発効率にこだわると，マシン語っぽさがなくなってしまうこともありそうだ。なんとなくでき損ないの高級言語を使っているような気分になるかもしれない。使い回しの効きそうなルーチンはがっちり作ってどんどんライブラリに加えるのと同時に，必要なら独立性は捨て，思いきり実行速度にこだわって趣味の世界に走るという楽しみ方も忘れてはいけないような気がする。

ということで，また来月。

8)　Human68kではライブラリファイルは拡張子“.A”で表す。

9)　MS-DOS上でX68000用のプログラムを開発するためのクロス開発ツールにはライブラリアンがあり，リンカもそれに対応しているそうだ。XCの次バージョンあたりではきっとライブラリアンが採用されることだろう。

10)　AR.XはC compiler PRO-68KかTHE福袋V2.0についてくる。ごめん。

.even疑似命令の意味

簡単にいうと，.evenはアセンブル時に次の命令なりデータなりが偶数アドレスに置かれるよう調整する働きをする。すでに次の命令が偶数アドレスに置かれる状態であればなにもせず，奇数アドレスに置かれそうになっていたら1バイトの詰めものを入れる。かたい言い方をすれば，“ロケーションカウンタの値を偶数境界に整合させる”命令である（ロケーションカウンタというのはアセンブラが次に命令やデータをどこに置けばよいのかを数えるのに使うもので，AS.Xの場合はセクションごとに別々のロケーションカウンタが用意されている）。

なんでこんな疑似命令があるかというと，68000ではメモリに対してワード単位，ロングワード単位でアクセスする場合には，そのアドレスが偶数でなければならないという制約があるからだ。多少いい加減だが，68000が16ビットCPUだからだと思っていいだろう。16ビット幅のバスには奇数アドレスは“切りが悪い”のである。無理に実行しようとすると“アドレスエラー”が発生する。

これは68000がプログラマに与えるほとんど唯一の“頭の痛い”制約である。これが同じ16ビットCPUの8086になるともう少し制限は緩やかで，偶数境界をまたぐワードアクセスは内部で2回に分けて行うことになっている。実行速度が低下するだけですむわけだ。

メモリアクセスがすべて対象なのだから，プログラムカウンタの位置から命令コードを読み込むときもまた例外ではない。68000の命令コードはワード単位になっているからふつうは問題にならないが，プログラムの途中に奇数バイトのデータを挟んだりすると，やはりアドレスエラー発生の原因になる。プログラムカウンタはいつも偶数でなければならないのだ。

ものは考えようで，万が一pcがあらぬところを指しても，1/2の確率でアドレスエラーに引っ掛かって止まってくれるという見方もできる。これにメモリモードの保護機能が加わり，暴走しない，してもダメージが少ないという安全なプログラミング環境が得られるというわけだ。

ローカルエリア

ローカル（local）というのはX-BASICやCなんかのローカル変数のローカルと同じ意味で，“局所的”と訳すことになっている。これと対になるのがグローバル（global：大域的）だ。どちらも（主として変数名やラベル名などの識別名の）通用範囲（＝スコープ）を表す用語だ。字面を見ればわかるように局所的といえばごく狭い範囲でのみ通用するという意味で，大域的といえばもっとずっと広い範囲で通用することを表す。“通用する”でわからなければ，“見える”とか“使える”とか“意味を持つ”とか“（論理的に）存在する”とか適当に読み換えてみるとよい。

ついでに挙げておくと，同じような場面で使われる（が，意味は違う）言葉に“静的”，“動的”というのがある。変数のように“実体（この場合一定量のメモリ）を伴うもの”が，ずーっと存在していれば静的，現れたり消えたりすれば動的という。

Cだと多少複雑になるからX-BASICでいうと，メインルーチンで宣言した変数は，プログラムの（ほとんど）どこからでも使えるから大域的であり，いつでも存在するから静的である。また，関数の中で宣言した変数と，関数の仮パラメータとして宣言した変数は，一部の範囲（宣言した関数内）だけで使えるから局所的であり，必要に応じて作られ，用がすんだら消される運命にあるから動的だ。

大域的な変数はふつう静的であり，動的な変数はつねに局所的である。ちなみに，静的でも局所的な変数はありうる。物理的にはずっと存在していても一部からしか見えなければ，やはり局所的と呼ぶわけだ。

で，linkのところで出てきたローカルエリアというのは，サブルーチンの中で一時的に使用するために確保するメモリ領域をいう。linkを実行した時点で確保され，unlkで解放されるわけだから，動的でかつ局所的である。

ローカルエリアを用いる第1の利点は，一時的な作業用にわざわざ固定のメモリ領域を用意しなくてもすむということだ。linkではスタック上にローカルエリアを確保するわけだから，結局は同じメモリを複数のサブルーチンで首尾よく共有することができる。同じメモリ領域を使うとはいっても確保した範囲はその瞬間にはほかのルーチンで使っていないことが保証されているから（スタックの未使用部分をこっそり使うような悪いプログラムでない限り）ぶつかり合いを心配する必要もない。

また，第2の利点としては，ほかのサブルーチンに“余計なことを教えないですむ”ということが挙げられる。一時作業用メモリのような“自分だけが知っていればよいこと”をほかのサブルーチンにわざわざ教えてあげたり，“見える”ところに置いておく必要はないのだ。ほかのルーチンを“無知”にすることで自分の独立性を保ち，相手の独立を促すのである。

さて，図Aにスタック上にローカルエリアを確保する様子を示す。基本的にはスタックポインタを強引に移動して空間を作ってやるというだけのことだ。

図のa）の状態から，

```
subq.l  #12,sp
```

とか，

```
lea.l   -256 (sp),sp
```

のようにスタックポインタを直接操作し減じれば，それぞれ12，256バイトのローカルエリアが確保される。b）を見ればわかるように，確保したローカルエリアはsp自身によってポイントされることになる。また，c）はローカルエリア確保後，スタックにデータを積んでもローカルエリアは破壊されたりしないことを，最後のd）はローカルエリアを解放するには，spを“ローカルエリアを確保する前の値”に戻してやればよいことを示している。

このように，link命令を使わなくてもローカルエリアは確保できるわけだが，linkの旨味は確保したローカルエリアをスタックフレームに取り込めるという点にある。図Bにlink命令を使ってスタック上にローカルエリアを確保した様子を示す。これは，

```
link    a6,#-8
```

によって8バイトの領域を確保している。

a）の状態でlinkを実行するとb）のようになる。直接spを操作したときと同様，この時点では確保したローカルエリアはspによってポイントされている。フレームポインタであるa6を基準にすれば“－8（a6）”で示されるアドレス以降がローカルエリアである。

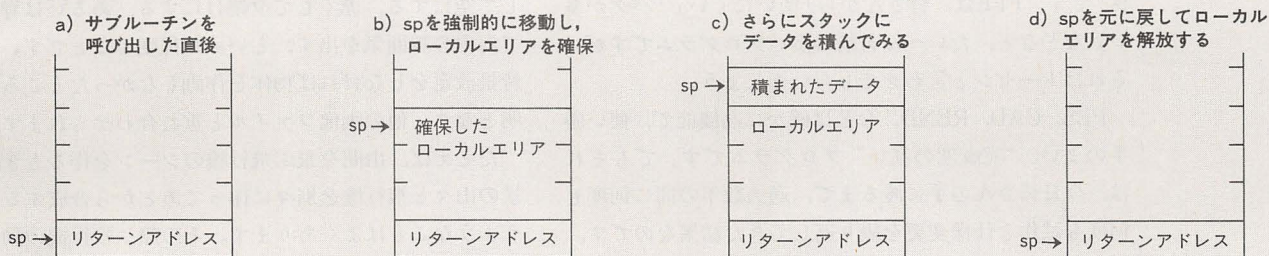
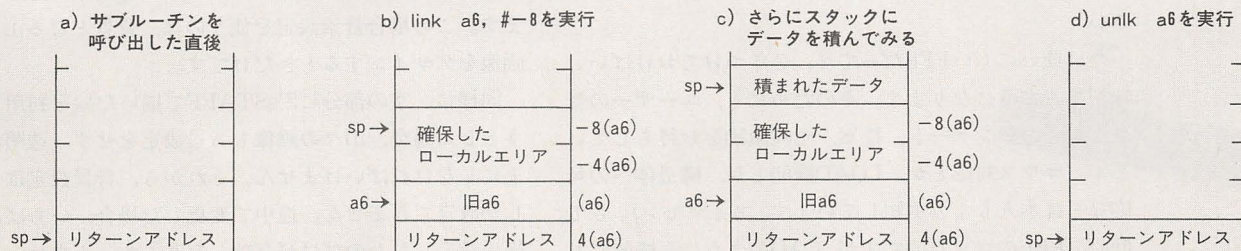
c）はb）の状態からロングワードデータひとつをスタックに積んだ状態だ。spはすでにローカルエリアを指していないが，フレームポインタから見たローカルエリアは同じ位置“－8（a6）”を保っている。

最後にlinkを使ってローカルエリアを確保する際の注意点を挙げておく。

上記の使用例を見ると気づくように，linkの第2オペランドは“－（マイナス）確保するローカルエリアのバイト数”になる。68000のスタックはアドレスの低いほうに成長していくから，ローカルエリアを確保するにはspの値を小さくしてやらなければならないわけだ。よって，linkの第2オペランドは必ず0か負の数になる。

また，第2オペランドは16ビットの符号付き数として扱われるので，指定可能な範囲は－32768～＋32767までになる。が，正の数は使う意味がないので実際には－32768～0の範囲のみが使われる。このことはlinkで確保できるローカルエリアの大きさが最大32Kバイトまでであることを意味している。

さらに，68000の謎の制約により，第2オペランドはつねに偶数でなければならない。奇数にするとアドレスエラー攻撃を受けることになる。これに関しては前ページの『.even疑似命令の意味』を参照してもらいたい。

図A　ローカルエリアの確保

図B　link，unlkによるローカルエリアの確保・解放

「くさってもFFE」

プロジェクトチーム DōGA かまた ゆたか
三保 陽介

FFE（フレームファイルエディタ）は動きのデザインツールです。完成予想図を見ながら動きのデザインができるというメリットがあるため，使用頻度の高いツールですが，CGAシステムの中でももっとも使いにくいとの定評もあります。そこで，開発者の三保君自らFFEの使い方のコツを公開してもらいましょう。

こたつの中でマウスをコロコロする季節になりましたが，CGAシステムは元気に働いているでしょうか？ CGAシステムの応募は先月末で締め切りましたが，どうしても！ という方には，まだ対応します。しかし，今から申し込んでもいつ発送できるか見当がつきませんし，実費も若干高くつきます。またマニュアルがなくなった時点で打ち切りということもありますので，その節はご了承ください（カンパなどはお返しします）。

さて，今月から3カ月連続で動きのデザインについて取り上げようと思っています。今回はまず手始めということで，FFEの使い方です。私自身は，まだあまりFFEを使いこなしていないこともあり，これから3カ月の長丁場に備えて体調を整えるという意味で，そろそろ三保君にバトンタッチしようと思います。それでは後半の「こんなときどうする？」増刊号で，またお会いしましょう。さよなら，さよなら，さよなら。

FFEの使い方

ということで，強引に原稿を押しつけられた“CPU 三保”です。FFEは，皆さんから「使いにくい」「バグが多い」などなど，たいへん苦情の多いプログラムですが，それはトーゼンと言わせてもらいましょう。

PES，CAD，REND，などは確かに高機能で，使い勝手のよい，“完成度の高い”プログラムです。でもそれは，今日皆さんの手に渡るまで，過去数年の間に何度も何度も試作と仕様変更を繰り返してきた結果なのです。FFEは今回のCGAシステムの配布のために，取りあえず用意したプログラムなのですから，一緒にしないでください。

今は使いにくいFFEだって2，3年つけておけばいい味が出るようになります。そのためにも，ユーザーの皆さんからのアンケート，提案，叱咤激励をお待ちしています（マウス対応とか，LOAD機能とか，構造体への対応などは本人も十分承知しています。スイマセン）。さて肝心のFFEのコツなのですが，CADのように高機能でもないし，アトリビュートのようにユーザー側のノウハウの蓄積もないし，マニュアルの第3章でも詳しく解説されているので，実はあまり書くことがないのです。要はFFEをだましだまし，うまく使ってやろうということです。

●FFEを始める前に

まずは，紙を用意してください。えっ，なぜかって，それは，その紙に簡単に動きを描いておくからです。あなたが今アニメーションを作るとしたら，どんな動きにしますか？ 車のレースとか，ボールが跳んだり跳ねたりするものですか？ いきなりFFEの画面を見ていても思いつくものではありません。まず，どの物体がどのような動きをするのか，ラフスケッチでいいですから，紙に描いたほうがラクになるのです。マニュアルを持っている人は，第3章32ページによくわからない参考例が載っているので見てください。

では，実際にFFEを使ってみましょう。

●背景設定の使い方

背景設定などと大げさに言っていますが，現在のバージョンでは，ただ単に指定された色で画面全体を塗りつぶした上に作画するだけです。使い道としては，水色にして空にする，赤くして夕焼けにする，あるいは青くして海中の雰囲気を出す，といった程度のことです。逆に背景設定をしなければ物体を作画しなかったところは透明となり，他の画像ファイルと重ね合わせられます。

たとえば，山間を飛ぶ飛行機のシーンを作るとき，背景の山々と飛行機を別々に作ってあとから合成するというようなことはよくあります。その際，飛行機の動きをデザインするときに，バックを背景設定で空の色にしてまうと，もう山々の画像ファイルとは合成できなくなります。この場合背景設定を使うのは，背景となる山々の画像をデザインするときだけです。

同様に，空の部分にZ'sSTAFFで描いた絵を利用しようとした場合，山々の画像も背景設定をせず，透明のままにしなければいけません。それから，背景設定は1回しか設定できません。途中で変更した場合，いちばん最後に設定したものだけが有効となります。1フレーム目は青色，100フレーム目は赤にして，だんだん夕焼けにな

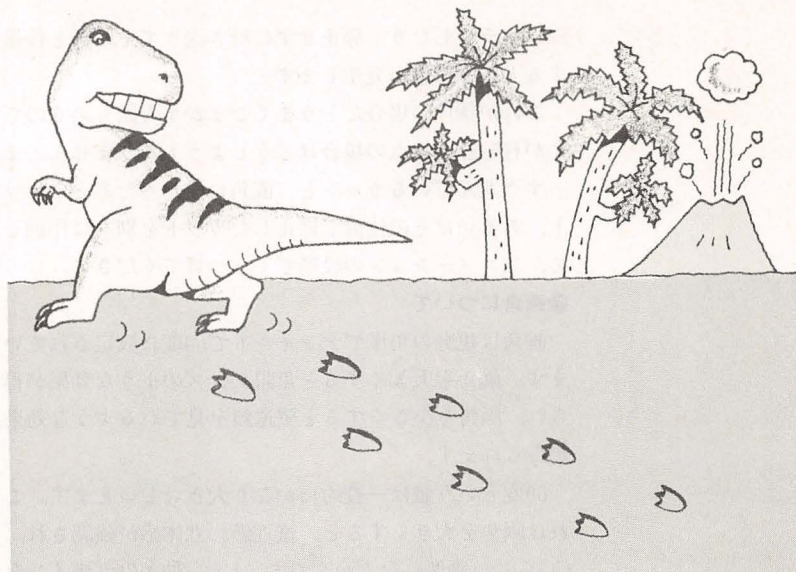
っていく様子を表現するなんて器用なことはできないのです。おあいにくさま。

●光源の設定

RENDでは，平行光線，点光源を複数配置でき，ライティングも凝ったことができるのですが，FFEではまだサポートしていません。光源の設定はベクトルなども与えないといけないため，めんどうくさがってデフォルト（なにもしなければ自動的に与えられる値）にしている方が多いようですが，物体の見ばえがいいようにするためには大切なことです。きっちり与えましょう。

光源のデフォルトの向きは，（−3，−2，−4）となっています。これは物体の正面から見たとき，つまり物体よりX座標が大きな点に視点を設定したときに，物体に当たる光がちょうど右斜め上から差し込んでいるようになります。この右斜め上から光を当てるのは，ライティングの最もオーソドックスで自然なやり方です。ですから，物体が原点付近にあり視点がX軸（この場合Y，Z軸でも同様）の負側にあるのに光源設定を直さないと，やけに暗い絵になって，凹凸もわかりにくくなってしまいます（いわゆる逆光の状態）。視点がX軸の負の側になっているのなら，光源のベクトルはX軸の正の向きにして（3，−2，−4）とでもすればよいでしょう。

また，中心となる物体が飛行機や宇宙船といった浮遊物である場合，光を斜め下から当てると，浮遊感が強調されます。ベクトルのZ軸を正にして（−3，−2，4）のようにします。

次に光源の色ですが，これもなかなか表現力に味をつけるものです。夕焼けを表現したいときは赤く，海中なら青くします。色の与え方はアトリビュートと同じRGB方式です。0から1まででですので，1以上を入力しても1になります。夕焼けを表現するのに，光源の色は白のままで物体の色（アトリビュート）をあらかじめ赤くする方がいらっしゃいますが，これはいけません。ほかのシーンでの流用がやりにくくなりますし，夕日に照らされていない影の部分まで赤くなってしまいます。

先月号でも少し触れましたが，色を使いすぎて画面全体がまとまらないようなときも，光源にちょっと色をつけることで雰囲気を整えることもできます。この手法はちょっとポイントが高いですよ。

それから，光源も1回しか設定できません。途中で変更した場合，いちばん最後に設定したものだけが有効となります。1フレーム目は青色，100フレーム目は赤にして，だんだん夕焼けになっていく様子を表現するなんて器用なことはできないのです。ゴメンネ。

●視点は急に止まらない

ここで，FFEの欠点をまたひとつ暴露してしまいます。FFEは動くものすべてをスプラインによって補間してしまいます（ただし，与えられた点が2点の場合は直線補間と同じ意味になります）。スプライン補間についての詳しいことは昔のOh! MZに任せるとして，とにかくここでは「指定された点を滑らかにつなぐ」と理解してください。そのために，まっすぐ進んで急に直角に曲がるとか，その位置でぴたっと停止することができないのです。FFEで上記のように設定すると，曲がりきれずにオ

アマチュアCGAコンテスト事務局より

第2回「アマチュアCGアニメーションコンテスト」の締め切りまであと1カ月ですが，皆さんの作品制作は順調ですか？　現在，正式にエントリーしている作品はまだありませんが，参加作品の制作のうわさはかなり入っています。皆さんもぜひ頑張ってください。

なお，このコンテストでは，鳥取大学の「ART-2」のように，DōGA CGAシステム以外で制作された作品も大歓迎ですし，もちろん機種は問いません。詳しい応募規定などは，先月号をご覧ください。当チームまで封書にてご連絡いただければ，応募用紙（マニュアルに掲載されているものと同じもの）をお送りいたします。

さて今回は，この連載の読者に限って，こっそり入賞のコツをお教えしましょう。まず，このコンテストでは作品としての完成度がある程度要求されます。昨年度のレベルを考えても，“CGAシステムを使い，とりあえず数カット作ってつなげてみました”的な作品はまず選外となるでしょう。

特別審査員の方々は，皆さん好みに差がありますので，“こういった作品は有利だ”という傾向は特にありません。野地先生は女性だけあって感性に訴える芸術的なものを好まれるだろうし，「PIXEL」の河内編集長はCGの専門家だけあって技術力を鋭く見抜きます。私は映像作品としてのストーリーや構成などに結構こだわるつもりですし，Oh! Xの前田編集長は“はったりでもいい，インパクトのあるもの”が好きなんだそうです。さらに，オリジナリティ，新しいアプローチ，単なる努力と根性も評価されます。また，一発ウケ狙いでもよいと思います。パロディについては，単なる人マネを越えていれば十分オリジナリティがあると受け止めています。

来年2月末に東京市ヶ谷で，3月初めに神戸三宮で，入選作品の上映会を行います。当方の希望としては全国各地で行いたいのですが，会場の用意ができません。日曜日に一般の方も入れるような会場（教室や市民会館？）を手配してくださる方いらっしゃいませんでしょうか？心当たりがある方は当チームの「会場に心当たりがあるんだけどなぁ」係まで。

ーバーランしたり，停止せずに行き過ぎてそのあと後進するという事態が発生します。

これが物体の場合だとうまくごまかす方法もあるのですが(後述)，視点の場合はどうしようもありません。まっすぐ進んでいるカットと，直角に曲がったあとのカット，あるいはその位置で停止したカットを別々に作画して，アニメーションの段階でくっつけてください。

●画角について

画角は視野の角度でデフォルトで60度に設定されています。画角を大きくすると魚眼レンズのような効果が得られ，画角を小さくすると望遠鏡を見ているような効果が得られます。

60度という値は一般的にかなり大きいといえます。これは画角を大きくすると，遠近感，立体感が強調され，CGらしい画像になるからです。また，動きのデザインを紙の上でするとき，視野を表すのに三角形が使えるというのも大きいメリットといえます。

しかし，自然な感じを出すときはもう少し小さくして，40度ぐらいにしても問題はありません。逆に，迫力を増すために画角をもっと大きくすることもありますが，だんだん不自然な絵ができるので，75度が限界でしょう。

また，ズームアップのような効果を得るために，1フレーム目は70度，と100フレーム目は30度にしておいて，滑らかに画角を変えていくということもできます。

●画面回転の使い方

画面回転とは，たとえば首をかしげた状態で見ているようなもので，視線を軸として画面全体が回転します。この画面回転を時間的に変化させてぐるぐる回すようなことは，ドッグファイトなどのアクションシーンぐらいにしか使いません(見ているほうが疲れてしまいます)。しかし，一定した値で少し（±15度）ぐらい傾けていてもあまり違和感がなく，構図の点でちょっとしたアクセントとして意外に効果があります。でも，±どちら側にどの程度傾けるかということは，高度なセンスを要求されるので，上級者向きのテクニックと言えるでしょう。できた画像がなにか物足りないといったときには，試してみてもよいのではないでしょうか。

●スケールを変更する「＋」，「＝」

マニュアルには隅っこに書かれているのですが，このキーは結構使えます。「＋」キーを押すと，原点を中心として平面図と側面図の見える範囲が，2，5，10，20倍に広がりますので，広範囲に渡って動くデザインのときに利用します。逆に「＝」を押すと，1/2，1/5というように，原点付近のアップとなります。細かい設定をするときに真価を発揮します。このようにして平面図，側面図の見える範囲は，最低ではX軸Y軸ともに±320，最大では±32000までとなっています。

●木を植えるときの「/」,「＊」

たとえば木を植えるときは，通常地面(Z座標が0)に根を置き，Z軸座標が正の空間に幹を伸ばします。つまり，Z座標の負の領域をまったく使用しないわけです。家や車の場合も同様です。

FFEの側面図は中心が原点になっていますが，これでは下半分が無駄になってしまいます。そこで，「/」キーを押すと，側面図の原点が下がり，正の領域だけを表示するようになります。そして「＊」キーを押すと原点が中央に戻ります。

この「＊」「/」「＋」「＝」キーですが，これらはいつ

寺田の教育的指導・世間は広い！

先月はお休みしてしまい，申し訳ありませんでした。ちょうど大学の試験期間と重なって私は身動きが取れなくなってしまったのです。

さて，私が試験で死んでいる間にも，続々と作品が届きました。これまでにもいろいろと妙な手紙やハードディスク（！）などが送られてきて驚いたことがありましたが，また違った意味で驚いてしまったのがこの方，なんと「自衛隊・第三輸送航空隊・修理隊・油圧分隊(おお！なんや知らんけどすごい！)」勤務の村上公三さんです。自衛隊勤務で，X68000持ってて，そしてCGAシステムを申し込んだ人，こんな「理想の結婚相手」(？)のような人がいるとは！　やはり世間は広いんですね。

肝心の作品ですが，村上さんの日常生活を描いた作品「美保基地（米子空港）上空を飛ぶF86Fブルーインパルス」，「飛び立つKV107」です。村上さんが整備してらっしゃるというC-1輸送機もちゃんと写っています。ブルーインパルスはいかにもブルーインパルスといったスピード感のあふれるものになっていますし，KV107も妙にリアリティがあります。CGAシステムを使い始めてすぐに，これだけの動きをデザインできるということは村上さんはかなりのCGAゴコロを持った方なのでしょう。

村上さんはもうひとつ，いかにもOh!Xの読者が喜びそうな（？）作品を送ってくださいました。これはなかなかかっこいいロボットで，うちのアニメファンも感心していました。私はよくわからないのですがこれは何かのアニメに出てくるものなのでしょうか？　アニメファン氏は見る人が見ればわかるといっていましたが……。

ともかく，とてもモデリングのセンスがいいのでこんな揚げ足を取るようなことを言いたくないのですが，ところどころ面と面の間に隙間があるようです。たぶんCADの「最近点」機能をまだご存じないのだと思います。CADで何も考えずにどんどん面を作っていくと，そこらじゅう隙間だらけになってしまいます。面の横につなげて面を作るときには，原則として既存の点を出発点にしてください。具体的には，次の面を作り始める既存の点の近くに3Dカーソルを持っていって，「最近点」とやります。すると，カーソルは正確に既存の点のところへ移動しますから，そこで「点確定」して，あとはいつものように次々と頂点を入力していけばよいのです。もちろん，より多くの既存点を使って面を作ることもあります。その場合はそのつど「最近点」を使ってください。こうすれば面と面がぴったりつながります。それと，送られてきた静止画は立体感が少し不足していましたので，私が簡単に手を加えてみました。まず，アトリビュートのアンビエントを少し小さくしました。また，光は平行光線のみでしたので，点光源をロボットの左斜め上に置いてみました。あと，せっかくの細かいモデリングなのでRENDで作画する際に512ドットモードにして，2倍アンチエイリアシングをかけてみました。元の絵と比べてみてください。

全体的に見れば，初めてでこれだけのものを作られたのですから，これは「技あり！」ですね。あとはもう少しアトリビュートを研究してみてください。

今年もCGAコンテストを開く予定ですが，いろいろと送られてくる作品を見ていると，昨年をはるかに上回る盛り上がりになりそうな気がします。これからも皆さんの意表をついた攻撃をお待ちしています。それではまた。

でも使用可能です（もちろん出力ファイル名の入力時などは除く）。特に物体設定のところで役立つでしょう。あと注意事項として，物体選択モードでも使わないほうがよいでしょう。なぜなら，平面図と側面図の物体および視点や視野の線は，書き換えを行いますが，選択カーソルは書き換えを行いません。ですから，これらのキーを押すと，変なところを指してしまいます。ただし，そこでまた選択すると，選択カーソルは，正しい位置を指すようになります。

●中止のときは「ESC」

それぞれの機能のところにきちんと中止のメニューがありますが，ESCキーのほうが速いし，必ずひとつ前の状態に戻るはずです。また，ESCキーを押し続けていればメインメニューに戻りますので，次のフレームを設定しにいったり，終了させたりすることが手早くできます。カーソルで中止を選択しようとすると，慌てて作画や実行をしてしまい，よけいな手間がかかることがあります。

●数値入力は「リターン」不要

物体設定においていろいろな数値を入力しなければいけないのですが，ひとつ入力するたびにリターンを押すのはやめましょう。リターンを押すとその時点で平面図と側面図を書き換えます。(0，0，0）にあった物体を(100，100，100)に移すとき，X，Y，Z座標を入力するたび書き換えていたのでは，時間がかかってしようがありません。入力項目間の移動はカーソルキーで行い，すべてを入力したあとでリターンを押すようにしましょう。

●フレームナンバーは，1フレームずつ設定しない

50フレームあるカットを設定しようとしたとき，50回指定するようなバカなことは決してしないでください(FFの意味がありません)。スプライン補間は，指定する点を増やせば増やすほど滑らかさがなくなります。1回曲がるためには3点も指定すれば多すぎるぐらいです。

●直進させたいとき

直線的に動かすときは，最初のフレームと最後のフレームだけ設定すればよいのです。しかし，指定したポイントが3点以上あったとしても，すべてが直線上に乗っていれば当然直線となるのですが，指定する点の間隔に差があると，滑らかにスピードが変化するような動きが表現できます。

●直角に曲がりたいとき

たとえば映画「TRON」の中にあったライトサイクルのように，まっすぐ突っ込んで，ある点でカクッと曲がるような動きをしたい場合，そのままスプラインにかけてしまうと滑らかにしか曲がりません。

そこで，以下の手順のようにします。1）まず第1フレームで希望の位置に設定。2）第10フレームで，進みたいところに位置を変更。3）そしてそこで「削除」を選びその物体を削除。4）今度は，その位置に新たに同じ物体を追加。そのとき向き（Z軸を90度回す）を変えておくことを忘れないように。5）そして第20フレームで進みたいところに位置を変えれば，すべて終了です。このようにするとまず最初の10フレームの間直進し，そこで瞬間的に90度ターンをして，また10フレーム直進することができます。この手法は，滑らかにつながないために，その部分の前後でその物体を定義しなおしているにすぎないのですが，直角に曲がる以外にもいろいろ応用が効き，急停止や，曲線運動から直線運動に移ることなどができます。「削除」という機能は，今いるフレームのひとつ前のフレームまで存在するようにしていますので，こういうことができるのです。

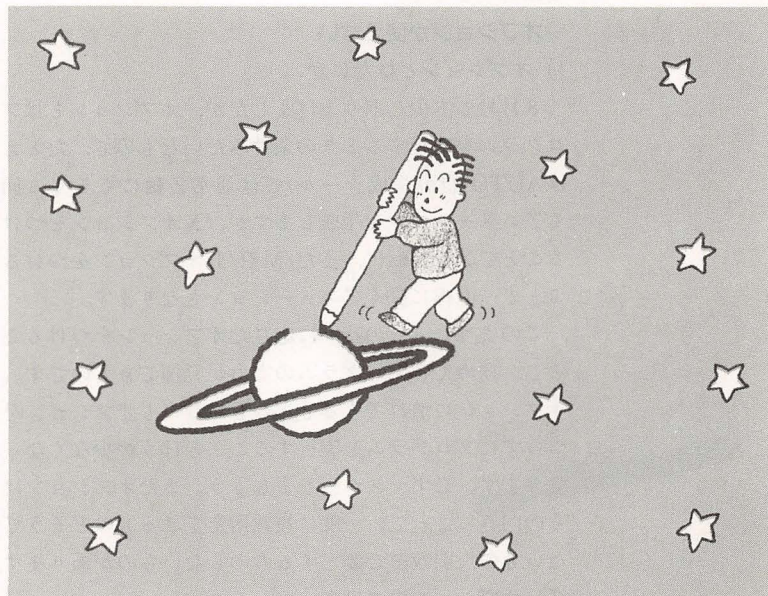
●フレームナンバーの設定は自由

フレームナンバーの設定は，最初は必ず1フレーム目から設定しなければいけませんが，小さいほうから大きいほうへ順番に指定する必要はありません。100フレーム目を設定したあとに，50フレーム目を用意することもできます。また，一度設定したフレームに再び入って，修正，確認を行うこともできます。

●似たフレームを流用する

新しいフレームを設定するときに，既存のフレームから変更することができます。たとえば，1フレームから90フレームまで10フレームごとに設定していたとします。次に100フレームを設定しようとしたとき，自分が作ろうと思ったフレームが，30フレームとよく似ていたとします。そのとき，まず30フレームを呼び出しそのあとで100フレームに設定すれば，30フレームの状態を基本として，100フレームを設定することができます。

＊

以上がFFEの使い方のコツです。FFEは，より使いやすくするため現在も改造中です。もちろん，自分でも努力はしていくつもりです（今の時点でもだいぶ改良されています）が，皆さんのご協力もお願いいたします。

「こんなときどうする?」増刊号

現在届いているたくさんのアンケートを見ると，皆さん同じようなトラブルを抱えていることに気づきました。今回は，特に初心者の方が“はまり”そうな問題について解説します。この連載中に，もう一度ぐらい特集をしたいと思っていますので，当プロジェクトルーム「こんなときどうする？」係へ，お便りお待ちしています。

◎大いなるかんちがい

1）スタートアップメニューが出なくなってしまった

スタートアップメニューはCGAシステムを使う前に，バックアップ，ハードディスクへのインストール，サンプルアニメーションの表示などを行うためのプログラムです。「CGAシステムを初めて起動したときにはちゃんと現れたのに，いつの間にか現れなくなってしまった。どうやら誤ってプログラムをひとつ消してしまったようなので，直してください」とのお手紙が何通かありました。

スタートアップメニューが起動するのは最初の1回だけです。以後は直接PES(CGA制作ウィンドウシステム)が起動しますが，安心してお使いください。

なお，もう一度スタートアップメニューを表示する必要ができた場合は，コマンドライン上から(CGAシステムを終了した状態から)，

B:¥>copy　a:autoexec.sta　a:autoexec.bat

とするだけで，入手したときの状態に戻ります。

2）データディスクを作るのにMKDATAを使う

MKDATAは，マニュアルの第3章「CGA制作入門」を実行するためのディスクを作るバッチファイルです。ですから，通常のデータディスクはごく普通にフォーマットされたものを使えばよく，MKDATAを実行する必要はありません。

◎オプションが使えない

1）オプションとはなにか

8月号でも少し取り上げましたが，オプションとはプログラムを実行するときの条件みたいなもので，たとえばAUTO(自動生成ツール)では通常Z軸にくるくる回るアニメーションを生成しますが，/Xオプションをつけることで X軸回転に，また/Mや/Lオプションをつけることで一直線に動くアニメーションとなります。

このように，その時々に応じたオプションをつけることで，初めて各プログラムの実力が発揮できるのです。オプションに慣れてくると，むしろなにもオプションをつけずにプログラムを実行することのほうが少なくなってきます。必ずマスターしましょう。またオプションはその内容によって，一度に複数指定できるものとそうでないもの，引数を必要とするものしないものがあります。

2）オプションのつけ方

通常RENDは1フレーム目から作画するのですが，作画中に中断したのち，その続きを再開したいといったケースはよくあります。ここでは具体的なオプションのつけ方として，REND(スキャンラインレンダラ)の/Sオプションで作画のスタートフレームを62，つまり62フレーム目から作画するように変更してみましょう。

PES（制作環境ウィンドウシステム）からRENDを呼び，必要なファイルを指定したあと，RENDのコマンド実行ウィンドウの右下の「オプション」をクリックします。すると右側にオプションウィンドウが開きます。/S（スタートフレームの指定）と書いてあるところをクリックしてください。マウスカーソルが消え，「/S［」の横にカーソルが現れました。ここに「62」を入力し，リターンすると再びマウスカーソルが現れますので，いつものとおり「実行」をクリックしてください。これで62フレーム目から作画します。

この例は，引数として数字を与えましたが，RENDの場合，/Oオプション（出力ファイル名の変更）のように引数として画像ファイル名（文字列）を与える場合もありますし，/N（画像ファイルを出力しない）のように引数を必要としない場合もあります。

なお，オプションの種類とその内容については，マニュアルの第5章をご覧ください。特に重要なオプションを挙げると以下のようになります。

AUTO	/V, /M
MIRR	/M, /O
OVLAP	/D, /O, /F
PILE	/O
REND	/A, /B, /C, /G, /O, /S, /T
SAVE	/L, /W
SLIDE	/O
SRANIM	/T, /M, /L

◎ちゃんと作動しない

1）CADのBGMが使えない

CAD（モデリングツール）はプログラム自体が大きく，またデータエリアも広く取る必要があるため，メモリが1Mバイトしかない方は，自動的にBGMの機能が削られてしまいます。

2）FFEでフレームファイルが出力できない

FFEの出力は，フレームソースとフレームファイルのどちらかを選択できるようになっています。といってもフレームファイルの出力は，フレームソースを出力して，子プロセスでFF（フレームファイルフィルタ）を実行しているだけです。ですから，CADのBGMと同様に，メモリが1Mバイトしかない方はフレームソースしか出力できないわけです。まぁ世の中，資本主義なんですわ。

もちろん，FFEを終了したあとFFを実行すればよいので，ほとんど問題ないでしょう。ここだけの話ですが，あるバージョンのRENDには，フレームファイルではな

くフレームソースを指定してもちゃんと実行してくれるという隠し機能がついているといううわさです。

3）CGAコピーでサンプルデータ集がコピーできない

バージョン2.00では正しくコピーできません。まぁちょっとしたミスというやつです（しつこくののしれ他人の失敗，笑ってごまかせ自分の失敗)。しかし，べつにプロテクトがかかっているわけではないので，diskcopyでコピーしても大きな問題はありません。

◎画質が悪い

1）ギザギザが目につく

アニメーションにするためには，256×256の解像度になるので，斜めの線がギザギザになるのは仕方がない……ことはありません。マニュアルの専門用語一覧の「アンチエイリアシング」をご覧ください。アンチエイリアシングは，境界線をぼやけさせることでギザギザを目立たなくさせる手法です。これは非常に効果があります。特にビデオテープに録画するときには，VTRに出力するときの画像の劣化が都合よくアンチエイリアシングの不自然さをカバーして，月と泥，雲とすっぽんの差となります。ぜひとも試してみてください。

アンチエイリアシングは，RENDで作画する際に/Aオプションをつけるだけで実行できます。/Aオプションには引数が必要ですが，これは何も考えずに「2」にしておけばよいでしょう。

しかし，世の中うまい話には罠があるもので，こんなにお手軽で効果的なアンチエイリアシングも時と場合によってはいろいろと不都合が発生します。

まず，作画スピードが半分になります。ですから，最後の仕上げのときだけアンチエイリアシングするようにしましょう。次に，生成される画像ファイルの大きさが倍近く大きくなります。ですから，ディスクに入る数も半分ぐらいになります。これは生成される画像ファイルの圧縮効率が悪いということなのですが，そうするとアニメーション再生時に1/10秒で展開できずに，カタカタとしたアニメーションになる可能性が高くなります。画像ファイルの大きさが25Kバイト以上になれば，まず間に合わないと思ってよいでしょう。

さらに，アンチエイリアシングをかけた画像は色数が3倍以上になるために，256色モードのアニメーション（「SRANIM実行時にメモリが足りなくなる」で解説します）ができなくなります。つまり，短くても高画質を希望するときはアンチエイリアシングをかけ，長時間のアニメーションを希望するときはやめたほうがよいでしょう。

2）高解像度（512ドット）で作画したい

これも簡単です。RENDで作画する際に，/Cオプションをつけるだけです。/Cオプションの引数は必ず512にします(256と512以外は無効となります)。低解像度と高解像度では，生成される画像の質が，猫と小判，花とだんごの差となります。さらにこの/Cオプションと/Aオプションを併用しようものなら，論よりツモ，猿も単位が落ちるという具合になります。

しかしながら，高解像度で生成したものはアニメーションにできませんのでご注意ください(SRANIMで再生はできますが，紙芝居的なスピードになります)。

◎メモリーが足らない

複雑な物体を作画させたり，長いアニメーションをしようとすれば，当然多くのメモリを必要とします。この際増設メモリを購入してもソンはない（備えあればうれしいな）と思いますが，拡張できないとあらば，メモリを占拠している不用なものを削除していけばよいのです。いくつかの場合に分けて解説しましょう。

1）たいていの場合有効となる方法

システムディスクにある「CONFIG.SYS」を書き換え

あき姫の迷える小羊のコーナー

こんにちわ〜，姫です。初めての試験シーズンも無事乗り越えましたが，なかにはそうもいかなかった先輩もいらっしゃるようです。

小羊：DōGAプロジェクトとプロジェクトチームDōGAの違いがよくわかりません。

姫：「手軽でパーソナルな映像表現としてのCGAの普及」を目指すプロジェクトがDōGAプロジェクトで，当チームはその参加団体のひとつというか，事務局みたいなものです。ですから，当チーム以外にもDōGAプロジェクトに参加している団体はいくつもあります。わかっていただけたでしょうか？

小羊：DōGAください。

姫：CGAシステムの配布はやっておりますが，当チームを譲るのは結構たいへんですよ。プロジェクトルームのマンション1室とパソコン，VTRなどの機材が多数，おまけにスタッフ数十人をつけて，かなりの金額になってしまうと思われるんですよねぇ……。

小羊：せっかくのコピーフリーなのにコピーしてあげる友達(X68000ユーザー)がいないのです。

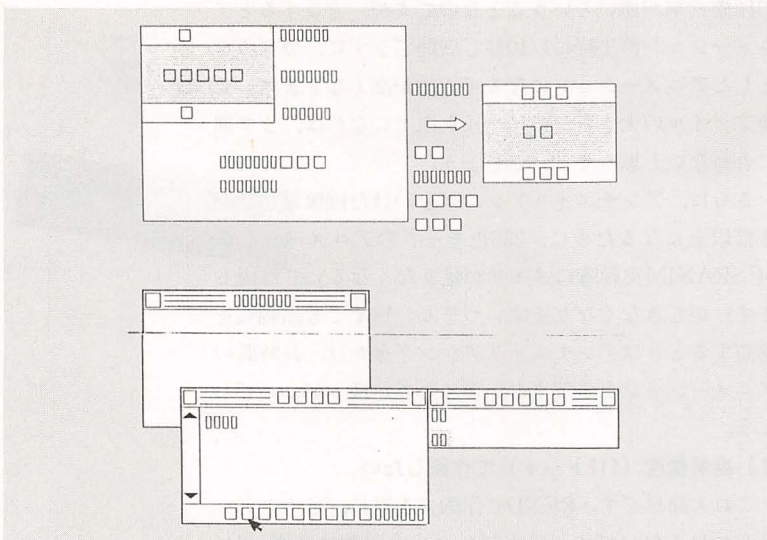
姫：PC-98ユーザーに「98じゃこんなことできないだろう」と見せびらかしてあげましょう。そのうちにX68000ユーザーになってしまうかも……。

小羊：ときどき「CRCエラー」というのが出ます。友人が言うにはフロッピーに傷があるのかもしれないそうなので，フロッピーの中の円盤を取り出し調べましたが，特に異常はありません。そのあとフロッピーを元どおりにしたのですが，以後まったくCGAシステムが起動しなくなってしまいました。どうすればよいのでしょう。

姫：……フロッピーディスクは大切に取り扱いましょう。

小羊：今，高校放送コンクールに出品する作品を制作しているのですが，CGAシステムで作った映像を，勝手に出品しても問題ないでしょうか？

姫：著作権保護法 第9条 「コンピュータのソフトウェアの保護と権利」に定めるところ，著作権の発生しているソフトウェアを使用しての著作物（以下第2次著作物と略す）に対する第1次著作物の権利については明確に保有するものとの判断がなされている……かなんてまったく知りません。コンテストに入賞すれば，CGAのよいPRになるし，DōGAプロジェクトの主旨にも一致しますので，基本的に大賛成です。

しかし，CGAシステムを使って制作した映像についての著作権を完全に放棄してしまうと，悪用されたとき（どのようにすれば悪用できるかは存じませぬが）にそれを差し止めることができなくなってしまいます。ですから，一応当チームにも著作権があるということにしておきますが，善良なOh!Xの読者は，全然気にする必要（当チームに承諾を得る必要も，入賞賞金を山分けする必要も）ありません。

ます。「CONFIG.SYS」がなんであるかはわかっている必要はありませんが，重要なファイルですのでお取り扱いは十分ご注意ください。

書き換え方は，ED（エディタ）で，普通のファイルと同様に「CONFIG.SYS」を読み込んで，

```
DEVICE　＝ ¥SYS¥ASK68K.SYS A:¥X68K_M.DIC A:¥X68 K_S.DIC
DEVICE　＝ ¥SYS¥PCMDRV.SYS
DEVICE　＝ ¥SYS¥OPMDRV.X #32
```

の3行の先頭に「*」を入れて，

```
*DEVICE ＝ ¥SYS¥ASK68K.SYS A:¥X68K_M.DIC A:¥X68 K_S.DIC
*DEVICE ＝ ¥SYS¥PCMDRV.SYS
*DEVICE ＝ ¥SYS¥OPMDRV.X #32
```

のようにします。ちゃんとセーブしたあと，リセットをかけます（「CONFIG.SYS」を書き換えても，再起動しなければ有効となりません）。こうすることによって，メモリのエリアはもう河童の質流れというぐらいに広くなります。

ただこのようにすると，漢字かな変換機能，AD PCM音源機能，FM音源機能はまったく使えなくなります（BGMを使うとエラーになります）。HDユーザーの方は特にご注意ください。

なお，「CONFIG.SYS」の「*」を取って再起動すれば，またもとの設定になります。

2）作画時（AUTO実行中）にメモリが足りなくなる

AUTOで作画せずにPESからRENDを使用するだけで，AUTOが使用しているメモリの分だけ，作画用メモリが増えます。かなり違うと予想されます。さらにPESを終了し，コマンドラインからRENDを使用すると，もう少しだけメモリが増えます。

3）SRANIM（アニメーション）実行時にメモリが足りなくなる

ビデオに落とす作品ならば2回に分けることができます。これはタイムチャートファイルを前半用と後半用の2つ用意すればすぐできるのですが，実際問題，ビデオで1フレームのずれもなくつなげるのは至難の技ですので，あまりお勧めできません。

簡単な方法のひとつに「.color 256」があります。これは色数を256色に制限することで，メモリの使用量を減らすやり方で，マニュアルの第6章7ページにあるように，タイムチャートファイルを書き換えます。

```
.timechart
.color 256              ←この1行を加える
      test [ 1 - 100 ]
.endchart
```

これだけで，使用量は半分になりますが，マニュアル

DōGA最新CGA作品紹介「Thank you VOYAGER」

ちょっとした手違いで，この作品の写真は先月号に掲載されてしまいました。本棚から先月号を取り出してから読んでください。

この作品はボイジャー2号の海王星最接近を題材にして，海王星の影に入って通信が途絶える32分間のボイジャーの孤独な格闘を描いたフィクションです。宇宙空間にぽっかりと浮かぶ海王星にボイジャーが滑り込むように飛んでいくオープニングタイトルなどはちょっと感動モノです。NASAの映像に負けないぐらいの……とまでいうと大ウソになりますが，パソコンのCGAシステムでも十分見るに耐えるものができることがわかっていただけるでしょうか。

この作品はあるコンテストに出品するため急きょ制作したものです（入賞すればオンエアされて，皆さんにもご覧いただけるかもしれません：取らぬ単位の胸算用）。CGAシステムの配布で忙しい合間を利用して2週間ほどで強引に制作しましたので，かなり手抜きです。なにしろ画面に登場する物体が海王星とボイジャーしかないのですから，構図などが似たカットの連続になってしまい苦労しました。最初に私が絵コンテを描き，イメージなどをスタッフ間で統一し共通の形状データを渡して，あとは各スタッフが好きなところから，自由に自分のイメージで作ってもらったのを集めて編集しようと思っていたのですが，ちゃんとカットを作ってくれたのは2，3人だけで，ほとんど私ひとりで作るはめになりました。

この作品を制作するに当たって，小さなプログラムを3つ開発しました（どれも1日でできる程度のものです）。ひとつ目は，512の画像ファイルから指定された位置の256ドットを切り抜いて256の画像ファイルを作るIC（イメージカット）です。切り抜く位置を連続的にX，Y方向に何ドットかずつずらしていくことができるので，背景が斜めにスクロールするような映像を作ることができます。2つ目は，STAR（スター）です。フレームファイルを入力すると，視点とターゲットの情報から，背景の宇宙空間を作画します（星の数も指定できます）。ボイジャーの画像とPILE（画像合成）してやると宇宙空間に浮遊している感じがよく出ます。最後のBOOM（ボム）はちょっと変わっていて，形状データを入力すると，各面をバラバラにした形状データを生成するとともに，指定された位置を中心に，各面が飛び散っていくフレームファイルを生成します。ちょうど爆発したような感じが出ます（まだ，かなり不自然ですが……）。これは最接近の際，海王星の表面粒子がボイジャーに衝突して火花を散らすシーンに用いられています。

このように，ちょっとしたプログラムで作品制作の作業がたいへん楽になったり，面白い表現ができるようになることはよくあります。皆さんもCGAシステムの開発を堅苦しいものと思わず，いろいろチャレンジしてみてください。

にある注意事項に気をつけてください。

もうひとつビデオに落とすときに限って使える有力な（こそくな）手段として，枠のPILEがあります。ビデオに出力するときは15kHzモードにしなければいけませんが，そうすると画面が縦方向に伸びて，上下20ラインぐらいが切れてしまいます。切れてしまう分のデータは無駄になるので，あらかじめ上下20ラインぐらいを黒のべたで塗りつぶしまうのです。

具体的には，まず「Z'sSTAFF」で枠（黒べたの部分）を描きます。黒にマスクをかけるとその部分が透明になるので，マニュアルの第6章71ページを見ながら画像ファイルに変換します（/Lオプション使用）。この枠をPILEでアニメーションしたい動画に重ねていくのです。

もう少しまともな方法として，/Mオプションがあります。/Mオプションの詳しい意味はパスするとして（マニュアルにはデフォルトが5になっているとありますが2の間違いです），1枚の画像データが大きいときは，3～6ぐらいにすると，メモリの使用量が減ることがあります。

まず，タイムチャートを書き換えて，そのアニメーションの一部だけを取り出しとりあえず実行します。ESCで終了する際に，画面に使用メモリ量が表示されますのでこれを記録します。同様に/Mオプションの値を変えてメモリの使用量をチェックすると，そのアニメーションの場合は，どの値で一番小さくなるかがわかります。

ちょっと話は変わって，この/Mオプションをどんどん大きくしてやると，メモリ使用量はどんどん増えますが，わざわい転じてフグと茄子，アニメーションの再生スピードが上がっていきます（/Tオプションとうまく組み合わす必要がありますが）。とても大きな画像ファイルを強引にアニメーションにするときは，メモリを犠牲にして/M50ぐらいにしてみるのもよいでしょう。

◎長いアニメーションができない

この問い合わせは結構ありました。「RENDの作画中にディスクがいっぱいになってしまう」，「SRANIM実行時に複数のタイムチャートを指定できない」，「複数のディスクにわたるアニメーションが実行できない」などいくつかの問題がありますので，順番にまとめて具体的に解説しましょう。

たとえば，「A」というカットが80フレーム（A001.PIC～A080.PIC）と「B」というカットが20フレーム（B001.PIC～B020.PIC）と「C」というカット30フレーム（C001.PIC～C030.PIC）を一度にアニメーションすることを考えます。「A」の作画中，62フレーム目で「ファイルがオープンできませんでした」というエラーで止まったとします。これは61フレーム目でディスクがいっぱいになってしまったということです。フォーマットした新しいディスクを画像データディスクとして用意して，「A＊.PIC」をコピーして，画像データディスクの「A062.PIC」を削除します。「A062.PIC」を削除するのは，このフレームは作画の途中だったので，正しいデータになっていないと思われるからです。そして，もとのワークディスクの「A＊.PIC」を削除します。これでワークディスクに空きができたので，続きを作画します。PESからRENDを呼び，/Sオプション（スタートフレームの指定）の横に「62」を入力し，いつものとおり「実行」をしてください。

今度は最後まで作画できたはずです。画像データディスクに「A＊.PIC」をコピーしてください。ところが今度はコピーの途中で画像データディスクがいっぱいになったとしましょう。「ディスクがいっぱいになりました」，「13のファイルをコピーしました」などと表示されるでしょう。画像データディスクを見ると，A001.PIC からA075.PIC までしか入っていません。このようなときは，もう1枚新しいディスクを用意して，A076.PIC ～A080.PIC を移してください。

以下同様に「B」，「C」のカットを作画したとします。「B」，「C」は2枚目の画像データディスクにちゃんと納まりました。次にタイムチャートファイルを作成して，1枚目のディスクに入れるのですが，1枚目はほとんど

ただいま会員募集中「CGAプロジェクトteam ART」

たびたび申し訳ないのですが，このコーナーの写真も手違いで先月号に掲載されてしまいました。レイトレで作画された，柱とガラスの砂時計の写真です。先月号をいったん本棚に戻してしまった方は，まことにお手数ですがもう一度取ってきてください。この場をお借りして，鳥取大学電子計算機研究会の皆さんにお詫び申し上げます。ごめんなさい。

さて，今回紹介しますのは，昨年の「アマチュアCGAコンテスト」にて優秀作品賞を受賞しました鳥取大学の電子計算機研究会です。受賞作品「ART-2」を今年の4月，新入生向けのデモに利用したところ，ぐっと新入部員が増えたそうです。コンピュータクラブに所属している皆さんもCGAシステムを有効に活用してください。

DōGA・CGAシステムをお持ちの方は，マニュアルの第7章の「レイトレーシング」の解説をご覧ください。そう，我々はCGAには向かないと断言（？）されているレイトレーシングによるCGAにあえて真っ向から挑戦しています。オリジナルのソフトウェア（レンダラー，モデラー，各種ツール類）から，ハードウェアの設計および制作（ACRTCを使用したビデオ出力可能高解像度フレームバッファ），それに生成画像コマ撮り用VTR編集機の自動制御までを網羅し，「ARTとしてのCGA」を目指す，自称芸術家集団です。

さっしのとおり，レイトレーシングはレンダリングに膨大な時間を費やしますので，夏休みなどの長期休暇中は合宿もどきを行って，作品制作およびハードウェア，ソフトウェア開発に取り組んでいます。

前回のDōGA主催「アマチュアCGAコンテスト」では，どういうわけか優秀賞をいただき，一気に"ハク"がついた当チームですが，それに恥じないような作品をどんどん制作していこうと思っております。

それでは皆さん，第2回「アマチュアCGAコンテスト」でお会いいたしましょう。

〒680　鳥取市 湖山町南4-101
鳥取大学教養部西田良平研究室内
鳥取大学電子計算機研究会
team ATR 代表　門脇　隆成

いっぱいになっているので，1枚目の最後の画像ファイル「A075.PIC」は2枚目に移して，1枚目に若干の空きを用意します。

以上の結果作成されました2枚の画像データディスクの内容は以下のようになっているはずです。

1枚目	A001.PIC	～	A074.PIC
2枚目	A075.PIC	～	A080.PIC
	B001.PIC	～	B020.PIC
	C001.PIC	～	C030.PIC

それでは，これらのアニメーションを一度に再生するタイムチャートファイルを作成してみましょう。くわしいタイムチャートの書式についてはマニュアルの第6章「高度な使い方」5～8ページをご覧ください。

それではまず1枚目の画像データディスクを入れ，いつものようにMKTCH（タイムチャート作成）を実行します。「A.TCH」が生成しますので，エディタでご覧ください。

```
.timechart
    A [ 1 - 74 ]
.endchart
```

もちろんこのままでは，1枚目のディスクの画像を再生するアニメーションになるだけです。そこで，以下のように書き換えます。

```
.timechart
    .color 256
    A [ 1 - 74 ]
    A:A [ 75 - 80 ]
    A:B [ 1 - 20 ]
    A:C [ 1 - 30 ]
.endchart
```

もちろんこれは，0ドライブが「A:」に，1ドライブが「B:」になっている場合なので，HDユーザーの場合は注意が必要です。

さていよいよアニメーションの実行です。0ドライブにCGAシステムを入れ，1ドライブに1枚目の画像データディスクを入れます。そしてタイムチャートファイル「A.TCH」を指定して，SRANIM（アニメーション再生）を実行します。実行すると，0，1ドライブのアクセスランプ（ディスクの入れ口の上にあるランプ）が，しばらく交互に赤くなったあと，1ドライブのランプが赤のままになります。これは1ドライブからたくさんの画像データを読んでいるからです。この間に0ドライブに入っているCGAシステムを抜き，2枚目の画像データディスクを入れます（時間は十分あるので，慌てないでください）。タイムチャートファイルに記述したようにBドライブの74枚を読み終わると，Aドライブ（0ドライブ）に移って，読み込みを続けます。このようにして，2枚のディスクのアニメーションを行うわけです。

メモリが十分にあった場合，3枚以上のディスクからも同様に，1，0ドライブを交互に読み込むことで長いアニメーションを実行することができます。

◎使いにくい

1）CAD（モデリングツール）において，面呼び出しがめんどうくさい

面の数が多くなると，指定したい面にポイントを移すのに，初めから順番に面呼び出しをしていてはたまりません。そこで，よい方法があります。

3Dカーソルを指定したい面のある頂点の近くにもっていきます。「最近点」をクリック（テンキーでないほうの“8”のキーを入力したほうが操作性はよいと思います）すると，その頂点に3Dカーソルが移ります。「点－>面」をクリック（または，テンキーでないほうの“4”

サイクロン CG大会潜入記

プロジェクトチーム DōGA
モデラー高津

さて，レポーターを務めさせていただくのは，下っぱとして東京まで連れて行かれた不幸な1回生の筆頭，モデラー高津です。

10月7日に，アンス・コンサルタンツ主催「第1回 サイクロンCG大会」が渋谷のフォーラム8でありました。この大会は同社が販売しているレイトレーシングソフト「サイクロン」で作った作品によるコンテストです。審査員として，シャープの鳥居部長や本誌の前田編集長といった方々もいらっしゃっていました。

この大会に寄せられた作品は，当日の飛び入りを含めて27作あり，なかなかの力作ばかりです。しかし残念なことに，我がチームの注目するアニメーション部門への応募作品はひとつもありません。なにしろ作画に時間のかかるレイトレーシングですので，高速性を売りものにしているサイクロンとはいえ，応募期間が少し短すぎたのではないでしょうか。

私たちDōGAはアーマットとともに，アンス・コンサルタンツの協力会社ということで，招待されていました（しかし，DōGAは「会社」じゃないんですけどね～）。応募作品の紹介の後，審査発表までフリータイムということで，アーマットの「Z'sTRIPHONY」と並んで，ご存じDōGA・CGAシステムの展示を行い，最新作の上映なんかもしていたのです。アンス・コンサルタンツも，来年の初めに出すというサイクロンの新バージョンをデモっていましたが，このバージョンではポリゴンデータが扱える，つまりCGAシステムのデータやトリフォニーのデータが読み取れるというウわさです。

ではそろそろ応募作品を入賞した8作品を中心に，独断と偏見によって，いくつか紹介したい思います（写真は63ページ）。

まずは，グランプリの作品，「SALOON」制作：益津 享氏から。「どこかなつかしく，心休まる空間」を描いたというこの作品は，他の作品にはないひとつの「世界」が築かれていると思います。一見，やたらマッピングをしているだけに見えますが，単にデザインしたものを表示させてみたという域を越えた“表現”を持っており，完成度という点で，グランプリというのも順当ではないでしょうか。しいていえば，従来のレイトレのサンプル作品にも似たようなものが多く，新しい方向性というものが欠けているような気もします。

次に，モデリング賞の「BIKE」制作：橘元弘司氏。これはもうご苦労さまと言うしかありません。こんな複雑な物体をデザインするなんて，考えただけで頭が痛くなりそうです。まさにモデリング賞を与えるにふさわしい作品でしょう。よく見ると，タイヤの溝までバンプマッピングしているのですね。たぶん最初からモデリング賞を狙って制作されたのではないかと思いますが，できることならバックなどにも凝って，作品としての仕上げにも力を入れてください。

ちょっと趣旨を変えて，レンダリング賞の「ダンス」制作：田淵友章氏について。確かにこの

のキー）するとその頂点を含む面がポイント面になります。ひとつの頂点に複数の面がくっついていることがよくありますが，そのような場合，続けてクリックすることで，それらの面を順番に移っていきます。

2）BGMをやめてほしい

さすがに起動のたびに同じ曲がかかっては飽きてきますね。「AUTOEXEC.BAT」の，

copy a:¥music¥アカルクイコウヨ.opm opm>nul

の1行をEDで削除してください。すると起動時には曲は鳴りません（BGMで曲を選択すると鳴りだします）。同様に「アカルクイコウヨ」の部分をほかの曲名にすることで，その曲を鳴らすこともできます。

3）CGAシステムを再起動させたい

PESを終了した後，再び作業を再開しようとするとき，「B:¥>PES」としても，「コマンドまたはファイル名が違います」と表示されてしまいます。これは単純にパスが切れてないだけなので「A:¥DOGACGA」にもパスを通せば問題ありません。しかし，パスの切り方がわかっているという方は，こういった問題で悩まないでしょう。めんどうですので，CGAシステムを再起動するときは，

B:¥>A:¥DOGACGA¥PES

とすればいいと覚えてください。

おわりに

今回はFFEを使用して動きのデザインを行ってみました。表現に限りがあるものの，フレームソースファイルを直接書くよりはずっとわかりやすいでしょう。前回のコラムでも触れましたが，このようなツールを使っての動きのデザインは，これからどんどん盛んになってくることが予想されますので，FFEはその参考にでもなれば幸いです。そんな動きのデザインツールには，より簡単に，より自由度高く，より自然に，あるいは汎用的に，逆に特定の物体について専用的に，などさまざまなアプローチがあると思います。皆さんのアイデアで，ユニークなプログラムが発表されていくことを期待しています。

ところで，10月号の「CG大会潜入記」のなかで「極上魔人アジオージャ」という作品を紹介しましたが，制作したのは東京電気通信大学のアニメーション研究会ではなく，電気通信大学の漫画アニメーション研究会だったそうです。この場を借りてお詫びいたします。

さて，次回はついにフレームソースファイルの書き方です。これをマスターしないとCGAシステムの表現力のすべてを引き出すことはできません。しかし，かなり難しいものですので，以下の宿題を必ず行っておいてください（宿題を忘れた者は廊下に立たせます）。

・ＥＤ(エディタ)の使い方をひととおりマスターする。

・マニュアルの第６章の「フレームファイルの書き方」，「フレームソースの書き方」を見ながら，FFEが出力するフレームソースを解読してみる。

と，いうことで，わ〜い，わ〜い，にこ・にこ・ぷん！

★臨時ニュース

今回のコラムのなかでも取り上げました，DōGAの最新作「Thank you VOYAGER」が，このたびめでたく集英社主催「BJ映像フェスティバル」に入賞しました。うれしいことに，このコンテストに入賞した10作品をまとめたものが，全国約1万軒（！）のレンタルビデオ店にて「ムービーリーグ」と称して並びます。読者の皆さんにとっては，今まで写真（静止画）としてしか見られなかったCGAが，実際に動くところを見ることのできる絶好のチャンス！　なのです(それに,本連載中に同作品を例にした実戦編を掲載する予定です)。なお，店頭に並ぶのは，11月末から年内（？）と期間が短いので，要注意！　ですよ。

作品は，反射，屈折が多く，レンダリングが大変だったと思いますが，しかしレンダリング賞ってどういう意味があるんでしょう。レンダリリングしているのはサイクロンとX68000なのですから，……ま，いいか。

次は，Oh!X賞の「propose under the moon beam」制作：菊池 彰氏。この作品は２枚組になっていて，もう一方の「Lover in sky」もすっかり気に入ってしまいました。今回はスペースの都合で１枚しか載せられませんでしたが，機会があればぜひとも紹介したいですね。２枚だからというより，１枚１枚にストーリーが感じられるのがすごい！　テーマを持って作品を作るというのは本当に大切です。色使いといい，構図といい，きっとその方面の仕事をなさっているのでしょうね。でもそのプロらしさが，作品を型にはめすぎることにならないかという不安は感じました。

同様にまとめ方がすばらしいと関心した作品「TAMANORI」制作：駒切 正氏はSHARP賞を受賞しました。こういった作品は個人的に大好きです。

そのほかの受賞作品をまとめて紹介します。

SHARP賞　「ワキウリ」　制作：富田 保男氏
LOGIN賞　「CHILD」　制作：塚田 哲也氏
同　「レイトレックに負けるなよ」　制作：江畑 一氏

また，惜しくも受賞を逃した作品にも特筆すべきものはたくさんあります。

まずは，「スタースピーダー」制作：山崎 誠氏。山崎さんは「P-REPORT」（かまたさんが４年前に制作した門外不出の迷作８mm映画）のファンだそうで。うんうん，作品にその流れを感じます。

そして「欲望の適応」制作：原 志津雄氏。タイトルは難しいけど実際の絵はご覧のとおり，かわいいニンジンとウサギさんです。こんな作品も肩が凝らなくていいなぁ。

「幻想の銀河中心」制作：渡久山 朝賢氏。これは作画だけでも，10日以上かかっているそうです。ご苦労さまでした。でも計算する時間が長いこと自体は，決してよいことだとは思いません。

「惑星」制作：柳沢 学氏は神秘的な雰囲気が出て，なかなかかっこいいと思います。

「AVERAGE PEOPLE」制作：阿山 修氏。“没個性，コピー人間への批判”というテーマを取り上げたことは面白いですが，表現がいかんせん十分でなかったようですね。いくらテーマが大切でも，まず映像としての完成度をおろそかにしては意味がありません。今後の作品を期待します。

全体的にかなりレベルは高く，すぐにでもプロとして通用できると思われるような作品も多くありました。パーソナルレベルでのソリッドモデルで，あれほど細かな表現ができるとは感動ものです。皆さん短期間の間にサイクロンを使いこなしているのですね。

ちょっと前までは夢のようだった環境が，ハード，ソフトの両面で整いつつある現在，パーソナルCGがどのような方面に，どこまで伸びるのか？　そんな期待を抱かせる「サイクロンCG大会」でした。

なお，来年以降も引き続き，第２回，３回と行うそうですので，読者の皆さんも今から腕を磨きましょう。

★(で)のショートプロぱーてぃ その4

TETROCK

Komura Satoshi 古村 聡

今月はX1用のアクションパズルゲームです。“ショートプロ”というにはちょっと長めのプログラムですが，面白さは保証つき。では皆さん，風邪気味の(で)氏に励ましの投稿プログラムを。

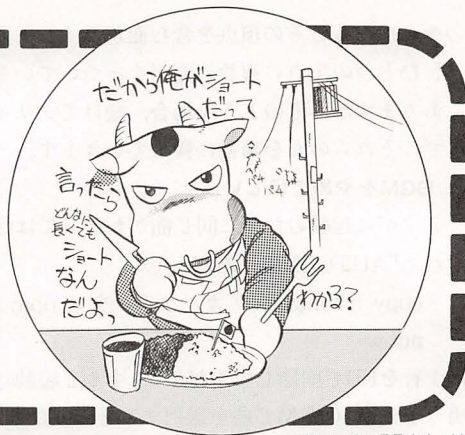



illustration:T.Takahashi

ぐずっ。びざざんごげんぎでずが？　がんべぎにがぜびいでびばっだ（で）でず。ずずっ（鼻がつまってぜんぜん会話ができない。「ぐずっ。みなさんおげんきですか？かんぺきにかぜひいてしまった（で）です。ずずっ」と本人はいっているんだけど……。ちなみに以下は訳した文です)。

ヅー，寒いのは嫌いだ。すぐ風邪ひいちゃうんだもんなー。いま私は鼻づまり，セキ，37度ちょっとの熱に加えて腹もこわしています。おかげで3日間なにも食べてないんだぞーっ。学校も休んじゃったし。さっき病院に行ってきたんですけどね。あの，病院というのも困ったもんですね。

うちの大学は総合大学だもんで医学部付属の病院（早い話が大学病院）があるので行ってみたんですが，まぁ凄い。なにが凄いってあの待ち時間。診てもらうまでに1時間，そのあと薬が出てくるまでさらに1時間……。診察は「あ，風邪ですね。……薬は食間に飲んでください」で3分かかってないですからねー，ヅーん。よけいにひどくなったんじゃないのかなぁ，風邪。大学病院っていえばもっと重病の人もきてるんだろうし……怖い考えになってしまった。ぐずっ。

マッハのスピードで壁を蹴る！

今月はX1用のワンキーアクションゲームなのですが，ショートプロの王道ですね，ワンキーアクションっていうのは。ボタン操作だけであるものを動かすってのは「どうやってワンキーで動かすか」というところに作る人のアイディアが生きるだけに面白いショートが多いみたいです。そういえば，昔ゲームセンターにあった（いまでもありますか？）ピンボールなんていうのも基本的にはボタン操作のアクションゲームですね（編注：ちょっと違わないか？)。

で，いきなり今月のプログラムに入るわけですが今月のプログラムはちょっとばっかしリストが長いです。マシン語のデータとかもありますしね。ま，長いだけのことはあるから（それに燃えますよ！　このゲーム）気合いで打つなり，読むなりしちゃってください。

これが噂のTETROCK

●TETROCK(X1/turbo)

宮城県　須賀　仁(18)

誰だっ，またTETRISもどきだろうなんていってるのはっ！　しっかり画面写真を見てから判断しましょう，見切り発進はとても危険です（編注：すみませんね。私は間違えました。でも，もう少しまぎらわしくないタイトルつけられません？)。

で，このプログラム，どういうゲームかというとTETRISも真っ青のワンキー反射型思考ゲームなのです。

まずルールから解説しましょう。面のパターンの中を自機の“<”がちょこまかと動き回りますからスペースバーでうまく操ってハートをすべて拾い集めてください。敵は4つの岩でこれに衝突すると自機が1機減り，全部なくなってしまうとゲームオーバー。そうそう，“■”を拾うと一定時間岩が動かなくなります。全部ハートを集めれば面クリア。スコアはハート1個につき10点×面数が加算されます。

で，このゲームのいちばんの特徴はなんといっても自機の移動です。スペースキーを押すことで自機の方向変換を行うわけですがその移動方向は，

壁づたいに進んでいるとき
……壁と反対方向に行く
壁に挟まれているか壁がないとき
……Uターンする

★毒物飲料探検隊日誌

○月×日　暴風雨　日直（で)

先月，私の所属する漫研の合宿で軽井沢に行ってきたんですけど，また探してしまいました。毒物飲料（な，なにしに行ったんだ，わざわざ軽井沢まで)。

全部で4種類（10本もありやんの！　どーりで重いと思った）買ってきたんですが，そのなかでもいちばん劇薬だったのが天下のUCCが販売している（ということは全国的に売り出されている可能性もあるわけだ！)「フィットネスペップ」。うちのサークルの連中も巻き込んで探しにいったんですけど，もう，これを見つけたときには「やったーっ，大当たりだーっ」と道の真ん中で叫んでしまいましたからねー。

ま，それはともかくこの「フィットネスペップ」，コピーが凄い。

「快・適・ラ・イ・フ・飲・料……フィットネスペップは高麗ニンジンエキス・ビタミン類などを含み，爽やかなグレープフルーツの香りで包まれた快適生活飲料です」

ひと口これを口に含むとグレープフルーツの爽やかで豊かな香りがノドを通り抜けたあとに，胃の底からでろでろと湧き出る漢方薬の絞りカスのような地獄の後味。げーっ，なんてサイバーな味なんだーっ！

これと一緒に買ってきたカゴメの「RIVELLA the unique drink」(ロ一杯に広がるハーブエキスの香り……えーい，うがい薬かおまえはっ！）もそうだけど，私は2度と飲みたくねいっ！というわけで読者プレゼントに回してしまいたいと思います。で，いちばん面白いことを書いてきた人にはショートプロぱーてぃ名物毒物飲料をさしあげちゃいますのでふるって投稿ください。ご意見，ご感想，ギャグ，イラスト，プログラムなーんでもありです。待ってまーす！

そいから，読者の方から持ち込まれた毒物飲料は一部プレゼントコーナーにまわされました。どーも，ありがとうございました。

で，テストプレイした感想なんですが，熱くなる熱くなる。操作が簡単（スペースバーを押すタイミングだけですから）なはずなんだけど……む，むずいっ！

頭の中では覚えているはずの移動方向(つまり壁があるときは壁と逆の方向……ってやつですね）に頭と指が追いつかないっっ！　うーん，あれは頭じゃなくてカラダで覚えるしかないのかなぁ，むむむ。ムズカシイけど面白い，面白いんだけどクヤシイ……，そんなゲームです。はい。

あ，そうだ。忘れてましたけど，EASYモードとHARDモードの違いは自機と岩のスピードが違うだけです。

元はX1turbo専用だったのですが，X1用BASICに移してSYMBOLやPLAY＠を殺してみたらすんなり動いたのでX1と共用に手直ししました。turboBASICで遊ぶ場合は，画面設定部分をそれらしく変えてKMODE 0を追加し，PLAY文をPLAY＠文に変えてください。よりオリジナル投稿版に近いゲーム展開ができます。

なんとなく，これだけではゲームの内容も面白さも伝えられたような気がしない(え，やっぱり？)のですが，このゲームばっかりは実際にやってみなければその内容も面白さもわかってもらえないような気がします。残念ながらこれはX1専用なのですがMZやX68000ユーザーの人もぜひX1ユーザーの人を見つけて実行して複数人で腕を競ってみてください。これはそのくらい面白いゲームなのです。

ああ，それにしてももうOh!Xに原稿を書きはじめて1年以上になるのに，本当に毎月毎月投稿プログラムの面白さを100％読者に伝えられたような気がしない……。才能ないのかなぁ。ただの2年目のジンクスだと思いたい……。風邪のせいか，考えがどんどん暗くなってしまった。ぐすっ。

リスト1　TETROCK

```
10 REM -------------------------  TETROCK  -------------------------
20 REM
30 INIT:CLEAR &HE000:CGEN:WIDTH 40:CONSOLE0,25:RANDOMIZE
40 CLICK OFF:REPEAT OFF:CLS4:FOR I=0 TO 7:PALET I,I:NEXT
50 PRINT"Wait a minutes.":PRINT:DIM PA$(15):HS=100:GOSUB1690:GOTO940
60 REM -------------------------  SCREEN  -------------------------
70 TEMPO500:CLS4:SC=0:ME=1:MA=5:HE=0
80 FOR I=0 TO 100:X=INT(RND(1)*112+207):Y=INT(RND(1)*200)
90 C=INT(RND(1)*7+1):PSET(X,Y,C):NEXT
100 CGEN:CLS:LOCATE 29,8:COLOR3:PRINT"HI-SCORE";
110 LOCATE29,3:PRINT"TETLOCK"
120 LOCATE 30,11:COLOR5:PRINT"SCORE";
130 LOCATE 29,16:COLOR6:PRINT"SCENE  ";ME;
140 LOCATE 29,19:COLOR2:PRINT"LEFT   ";MA;
150 LOCATE 27,23:COLOR4:PRINT"BY-HITOSHI.S";
160 LOCATE 30, 9:COLOR7:PRINTUSING"######";HS
170 LOCATE 30,12:COLOR7:PRINTUSING"######";SC
180 FOR I=0 TO 7:PALET I,I:NEXT:QME=ME
190 IF QME>7 THEN QME=QME-7:GOTO 190
200 ON QME RESTORE 1120,1280,1200,1240,1320,1160,1360
210 CGEN1:COLOR7:LINE(0,0)-(25,24),")",B
220 FOR I=1 TO 23:READ A$:FOR K=1 TO 6
230 H=VAL("&H"+MID$(A$,K,1)):LOCATE (K-1)*4+1,I
240 PRINT PA$(H);:NEXT K,I
250 POKE &HEF00,24,23,4:FOR I=&HEF03 TO &HEF0E:READ A:POKE I,A:NEXT
260 POKE &HEF10,0,&HAA,&HCC,&HF0,0
270 LOCATE PEEK(&HEF00),PEEK(&HEF01):COLOR7:CGEN1:PRINT"(";
280 FOR I=&HEF03 TO &HEF0C STEP 3:LOCATE PEEK(I),PEEK(I+1):PRINT"*";:NEXT
290 READ A:FOR I=1 TO A:READ X,Y:LOCATE X,Y:COLOR7:PRINT"#";:NEXT
300 FOR I=1 TO 20-HE
310 X=INT(RND(1)*24+1):Y=INT(RND(1)*23+1)
320 IF CHARACTER$(X,Y)<>" " THEN 310 ELSE LOCATE X,Y:COLOR7:PRINT"$";
330 BEEP1:BEEP0:NEXT:SOUND 7,&H38
340 FOR I=0 TO 100:A$=INKEY$:NEXT
350 PLAY"V12O4C5EGC+:V12O5C+GEC:V12O3C5GEG"
360 REM -------------------------  MAIN  -------------------------
370 IF HE=20 THEN 770
380 IF ST=1 THEN FOR K=0 TO 130:NEXT
390 IF PEEK(&HEF10)=0 THEN PLAY"V11O4C0G" ELSE PLAY"V11O6C0G"
400 IF PEEK(&HEF14)=1 THEN GOTO 800
410 DIR=PEEK(&HEF02):X=PEEK(&HEF00):Y=PEEK(&HEF01):BR2$=CHARACTER$(X,Y+1)
420 BR4$=CHARACTER$(X-1,Y):BR6$=CHARACTER$(X+1,Y):BR8$=CHARACTER$(X,Y-1)
430 IF INKEY$<>" " THEN 510
440 IF DIR<>2 AND DIR<>8 THEN 480
450 IF BR4$=")" AND BR6$<>")" THEN DIR=6:GOTO510
460 IF BR4$<>")" AND BR6$=")" THEN DIR=4:GOTO510
470 IF DIR=2 THEN DIR=8:GOTO510 ELSE DIR=2:GOTO510
480 IF BR2$=")" AND BR8$<>")" THEN DIR=8:GOTO510
490 IF BR2$<>")" AND BR8$=")" THEN DIR=2:GOTO510
500 IF DIR=4 THEN DIR=6:GOTO510 ELSE DIR=4:GOTO510
510 IF DIR<>2 THEN 560
520 IF BR2$<>")" THEN Y=Y+1:GOTO700
530 IF BR4$<>")" THEN DIR=4:X=X-1:GOTO700
540 IF BR6$<>")" THEN DIR=6:X=X+1:GOTO700
550 DIR=8:Y=Y-1:GOTO700
560 IF DIR<>4 THEN 610
570 IF BR4$<>")" THEN X=X-1:GOTO700
580 IF BR8$<>")" THEN DIR=8:Y=Y-1:GOTO700
590 IF BR2$<>")" THEN DIR=2:Y=Y+1:GOTO700
600 DIR=6:X=X+1:GOTO700
610 IF DIR<>6 THEN 660
620 IF BR6$<>")" THEN X=X+1:GOTO700
630 IF BR2$<>")" THEN DIR=2:Y=Y+1:GOTO700
640 IF BR8$<>")" THEN DIR=8:Y=Y-1:GOTO700
650 DIR=4:X=X-1:GOTO700
660 IF BR8$<>")" THEN Y=Y-1:GOTO700
670 IF BR6$<>")" THEN DIR=6:X=X+1:GOTO700
680 IF BR4$<>")" THEN DIR=4:X=X-1:GOTO700
690 DIR=2:Y=Y+1:GOTO700
700 A$=CHARACTER$(X,Y)
710 IF A$="*" THEN 800
720 IF A$="#" THEN POKE &HEF10,100
730 IF A$="$" THEN SC=SC+10*ME:HE=HE+1 ELSE 760
740 CGEN:LOCATE 30,12:COLOR7:PRINTUSING"######";SC:PLAY"O7C0GE"
750 IF HS<SC THEN HS=SC:LOCATE 30,9:COLOR7:PRINTUSING"######";HS
760 CGEN1:POKE &HEF00,X,Y,DIR:CALL &HE000:GOTO360
770 REM -------------------------  CLEAR  -------------------------
780 HE=0:ME=ME+1:POKE &HEF10,0:PLAY"O4C8R4:O4G8R4:O5C8R4"
790 PLAY"O4F8R4:O4A8R4:O5C8R4":PLAY"O4C8R4:O4G8R4:O5C8R4":GOTO100
```

```
800 REM -------------------------  GAME OVER  -------------------------
810 CGEN1:LOCATE X,Y:COLOR7:PRINT"!";:PLAY"R0"
820 SOUND 7,&H36:SOUND  8,&H10:SOUND  0,&HFF:SOUND  1,&HF
830 SOUND 6,&H1F:SOUND 11,&H8E:SOUND 12,&H5B:SOUND 13,&H0
840 LINE(X-1,Y-1)-(X+1,Y+1),"!",B:FOR I=0 TO 200:NEXT
850 LINE(X-2,Y-2)-(X+2,Y+2),"!",B:FOR I=0 TO 400:NEXT
860 LOCATE X,Y:PRINT" ";:FOR I=0 TO 200:NEXT
870 LINE(X-1,Y-1)-(X+1,Y+1)," ",B:FOR I=0 TO 300:NEXT
880 LINE(X-2,Y-2)-(X+2,Y+2)," ",B:FOR I=0 TO 500:NEXT
890 MA=MA-1:IF MA<0 THEN 900 ELSE 100
900 CGEN:FOR I=0 TO 7:PALET I,I:NEXT:FOR I=13 TO 24:PLAY"O6C1GEG:O4G1CEC"
910 LINE(25-I,25-I)-(I,I-1)," ",B:FOR J=0 TO 50:NEXT J,I
920 LOCATE 5,11:PRINT"G A M E  O V E R"
930 SOUND 7,&H38:TEMPO200:PLAY"V12O5B5O6CDEECO5BAABO6CDDO5BAGGAB+C+CAG#F#FO6EEDD
DDDO5B+C+DO6EECO5BAABO6CDDO5BAGGAB+C+CAG#FGAA4R1A5GGGG:V12O3R5RRC5O4G+C+EDA+C#F-
GDGB-EEGBCG+C+EDA+C#F-GG+C+G+FCDGC5O4G+C+EDA+C#F-GDGB-EEGBCG+C+EDCDGB+GBGD"
940 REM -------------------------  TITLE  -------------------------
950 CLS4:FOR I=0 TO 300:X=INT(RND(1)*640):Y=INT(RND(1)*200):C=INT(RND(1)*7+1)
960 PSET(X,Y,C):NEXT:CGEN1:COLOR7:LINE(0,4)-(39,14),")",B
970 LINE(1,5)-(38,13)," ",BF
980 RESTORE 1070:FOR I=1 TO 20:READ A,B,C,D:LINE(A,B)-(C,D),")":NEXT
990 POKE &HEF03,1,5,6,1,13,6,38,5,4,38,13,4,0,0,&HAA,&HCC,&HF0
1000 FOR I=0 TO 4:CALL &HE091:CALL &HE325
1010 CGEN:LOCATE 12,18:COLOR5:PRINT"EASY ····[1] key";
1020 LOCATE 12,20:PRINT"HARD ····[2] key";:FOR J=0 TO 20:GOSUB1050:NEXT J,I
1030 FOR I=0 TO 1:CALL &HE091:CALL &HE325:LINE(12,18)-(27,20)," ",B
1040 FOR J=0 TO 20:GOSUB1050:NEXT J,I:GOTO1000
1050 A$=INKEY$(0):IF A$="1" THEN ST=1:GOTO60
1060 IF A$="2" THEN ST=2:GOTO60 ELSE RETURN
1070 DATA 2,7,16,7,4,8,4,11,7,8,7,11,8,9,10,9,8,11,11,11,14,8,14,11
1080 DATA 18,7,18,11,19,7,21,7,21,8,20,9,19,9,21,11,23,7,23,11,24,7,26,7
1090 DATA 26,8,26,11,24,11,25,11,28,7,31,7,28,8,28,11,29,11,31,11
1100 DATA 33,7,33,11,36,7,34,9,35,10,36,11
1110 REM -------------------------  SCENE DATA  -------------------------
1120 DATA 000000,000000,3E3E7C,084010,10F820,210040,43E080,000000
1130 DATA 000000,000000,718E90,4A50A0,5250C0,6250C0,5250A0,498E90
1140 DATA 000000,000000,000000,000000,000000,0007FF,000000
1150 DATA 11,4,6,9,6,6,9,14,8,22,16,2,2,9,16,5,12
1160 DATA 001800,7FC3FE,7FC3FE,7F81FE,7FA5FE,7FBDFE,7E247E,7C813E
1170 DATA 79819E,7381CE,6799E6,0718E0,6799E6,7381CE,79819E,7C813E
1180 DATA 7E247E,7FBDFE,7FA5FE,7F81FE,7FC3FE,7FC3FE,001800
1190 DATA 1,1,2,1,23,8,2,12,4,24,1,2,4,11,1,14,1,11,23,14,23
1200 DATA 000C00,554CAA,000C00,554CAA,000C00,5540AA,000000,5502AA
1210 DATA 000000,550AEA,0100E0,512AEA,0700E0,512AEA,0100E0,550AEA
1220 DATA 000000,5502AA,000000,5540AA,000C00,554CAA,000C00
1230 DATA 1,1,6,1,23,6,7,12,4,7,14,4,3,11,8,11,18,5,13
1240 DATA 000000,7C8F7E,18A206,19A090,3DF080,3CF3AA,7E7300,437206
1250 DATA 4872B0,7E3202,06B202,10B2A0,02B3EA,12324A,00004A,3FE00A
1260 DATA 24E2AE,2022AE,0B0EEE,097E44,297000,3B3BBF,000000
1270 DATA 24,1,4,8,3,8,3,9,6,22,16,8,4,8,9,14,18,18,9,22,11
1280 DATA 000000,39F31C,444412,404490,38449C,044794,444492,384492
1290 DATA 000000,000000,000000,39139C,451110,411112,39F11C,051110
1300 DATA 451110,391390,000000,000000,0FC000,00007F,000000
1310 DATA 3,4,8,15,5,8,6,12,6,24,21,4,3,17,3,23,13,23,4
1320 DATA 000000,38661C,1C2438,0E1870,2718E4,3399CC,19DB98,0CC330
1330 DATA 065A60,0318C0,019980,00DB00,001800,01C380,0381C0,0724E0
1340 DATA 0E6670,1CC338,39819C,3300CC,260064,000000,000000
1350 DATA 3,1,4,21,1,6,11,11,2,19,12,4,3,3,3,4,21,22,3
1360 DATA 000000,400002,3FE7FC,3FC3FC,3F99FC,3F18FC,3E427C,3CE73C
1370 DATA 38001C,33FFCC,260064,0F81F0,00F780,30770C,38361C,3C363C
1380 DATA 3E227C,3F01FC,3F81FC,3FC7FC,3FE000,4000FF,000000
1390 DATA 1,1,2,12,4,4,22,2,8,8,11,6,5,11,7,16,7,15,12,18,13,11,17
1400 REM -----------------------  MACHINE LANGUAGE  -----------------------
1410 DATA 3E003214EFCD24E0CD70E23A14EFFE01C8CD91E0CD70E2CD25E3C93A00EF6F3A,34
1420 DATA 01EF67C9CD1BE03A02EFFE08C243E024CDFAE23E20CD86E0CD1BE0CDFAE23E25,30
1430 DATA C386E0FE06C25CE02DCDFAE23E20CD86E0CD1BE0CDFAE23E26C386E0FE02C275,C7
1440 DATA E025CDFAE23E20CD86E0CD1BE0CDFAE23E27C386E02CCDFAE23E20CD86E0CD1B,8C
1450 DATA E0CDFAE23E28444DED79C93E143215EFC93A10EFD601DA9DE03210EFC9CD8BE0,99
1460 DATA 3A03EF5F3A04EF573A05EF47CD2EE17B3203EF7A3204EF783205EFCD22E1CD8B,64
1470 DATA E03A06EF5F3A07EF573A08EF47CD2EE17B3206EF7A3207EF783208EFCD22E1CD,CB
1480 DATA 8BE03A09EF5F3A0AEF573A0BEF47CD2EE17B3209EF7A320AEF78320BEFCD22E1,9B
1490 DATA CD8BE03A0CEF5F3A0DEF573A0EEF47CD2EE17B320CEF7A320DEF78320EEFCD22,99
1500 DATA E1C9626BCDFAE23E2A444DED79C93A15EFD601C83215EF78FE02C284E114626B,DB
1510 DATA C5CDFAE2444DED78C1FE202813FE25280FFE26280BFE272807FE282803C371E1,E9
1520 DATA D5112800B7ED52D1C5444D3E20ED79C1C915CDC8E27CFE8038050604C384E106,74
1530 DATA 06C3D3E13A15EFD601C83215EF78FE04C2D3E11D626BC5CDFAE2444DED78C1FE,8D
1540 DATA 202813FE25280FFE26280BFE272807FE282803C3C0E123C5444D3E20ED79C1C9,DC
1550 DATA 1CCDC8E27CFE8038050602C32EE10608C322E23A15EFD601C83215EF78FE06C2,CA
1560 DATA 22E21C626BC5CDFAE2444DED78C1FE202813FE25280FFE26280BFE272807FE28,96
1570 DATA 2803C30FE22BC5444D3E20ED79C1C91DCDC8E27CFE8038050608C322E20602C3,19
1580 DATA 2EE13A15EFD601C83215EF15626BC5CDFAE2444DED78C1FE202813FE25280FFE,DA
1590 DATA 26280BFE272807FE282803C35DE2D511280019D1C5444D3E20ED79C1C914CDC8,45
1600 DATA E27CFE8038050606C3D3E10604C384E13A00EF2103EF96C287E23A01EF2104EF,09
1610 DATA 96C287E2C34CE33A00EF2106EF96C29EE23A01EF2107EF96C29EE2C34CE33A00,0F
1620 DATA EF2109EF96C2B5E23A01EF210AEF96C2B5E2C34CE33A00EF210CEF96C03A01EF,E1
1630 DATA 210DEF96C0C34CE3F5C5D5ED5BE2E2ED4BE5E2CDE7E222E2E2ED5F32E4E2D1C1,51
1640 DATA F1C989FA3683032100003E1029CB23CB12D2F5E2093DC2ECE2C9D5E506004C11,C1
1650 DATA 2800CD12E3545DE106004D2100301909D1C921000079B0C8CB38CB19300119CB,EA
1660 DATA 23CB1218F03A11EF07010010321 1EFE6FEED79043A12EF073212EFE6FEED7904,9D
1670 DATA 3A13EF073213EFE6FEED79C93E013214EFC900000000000000000000000000000,C7
1680 REM -------------------------  PCG  -------------------------
1690 DEFCHR$(32)=HEXCHR$("FCFCFC00CFCFCF0000000000000000000000000000000000")
1700 DEFCHR$(33)=HEXCHR$("648080000818100995A3CFFFF3C5A99997E7EE7E77E7E99")
1710 DEFCHR$(35)=HEXCHR$("FCAAC01AD983D500007E7E7E7E7E7E0000001824241800 00")
1720 DEFCHR$(36)=HEXCHR$("FCD8800081C3C70000247E7E7E3C180000247E7E7E3C1800")
1730 DEFCHR$(37)=HEXCHR$("FCFCFC3CFFEFEF0000000000000000000018183C3C666600")
1740 DEFCHR$(38)=HEXCHR$("FCFCFC1EDFFFEF000000000000000000000060781E1E786000")
1750 DEFCHR$(39)=HEXCHR$("FCFEFE3CFFDFDF000000000000000000000066663C3C181800")
1760 DEFCHR$(40)=HEXCHR$("FCFEFE78FFDFCF0000000000000000000000061E78781E0600")
1770 DEFCHR$(41)=HEXCHR$("FE828292838 3FF0000443838384400 00FEBAFEFEFEBAFE00")
1780 DEFCHR$(42)=HEXCHR$("FCC080008181C300003C7E7E5E6E3C0000000080400000000")
1790 RESTORE 1410:RR=2440:FOR I=&HE000 TO &HE351 STEP &H20:READ A$:SD=0
1800 FOR J=1,TO 32:ED=VAL("&H"+MID$(A$,J*2-1,2)):SD=SD+ED:POKE I+J-1,ED
1810 NEXT J:READ ZZ$:IF ZZ$<>RIGHT$(HEX$(SD),2) THEN 1850
1820 RR=RR+10:NEXT I:RESTORE1830:FOR I=0 TO 15:READ PA$(I):NEXT:RETURN
1830 DATA "    ","   )","  ) ","  ))"," )  "," ) )"," )) "," )))"
1840 DATA ")   ",")  )",") ) ",") ))","))  ",")) )",")))  ","))))"
1850 BEEP:PRINT"ERROR in ";RR:END
```

ACTIVE UNIT

X１用にちょっと変わったシューティングゲームをお届けしましょう。機動歩兵のパラメータを設定して，迫りくるミサイル群を迎撃してください。まわりは無重力の宇宙空間。慣性も考慮して行動しなければなりません。シンプルかつ高難易度の奥が深いゲームです。

Shibata Atsushi 柴田　淳

ところで

最近，不条理に難しいゲームが多いと思いませんか。

特にシューティングゲームに多いと思うのですが，弾がとにかくバラバラ出てきていつのまにかやられているといったようなもの，また，安全地帯とか必勝法がわからないと解けないようなゲームとか。

とても綺麗な背景にはセンスも感じられますし，大きなキャラクターがギンギンに動きまわるといった技術力は認めて余りあるのです。しかし，ゲームをやっていても「どうしてここが抜けられなかったのか」ということがわからずに，結局途中で投げ出してしまうようなゲームの不親切を僕は認めたくないのです。

いわゆる「ゲームのうまい子供」のプレイを見ていると，彼らは瞬間瞬間の論理的な思考（ゲームプレイヤーにとっては至極の芸術）によらず，「ここをこうやっていれば死なずにすむ確率が高い」といった，いつ偽りに転ずるかもわからないようなことに沿っているだけのような気がします。

そういった，すれっからしのプレイヤーの要求と彼らを消費者として崇めてやまない開発者の供給はお互いの相乗効果でとめどころのないほど膨れあがっていきます。本来，画面の美しさは付加的なものではなかったのでしょうか。弾の量は技術力に比例して増えるものではない，のではないでしょうか。

もっと単純なシステムで面白いゲームはできないのでしょうか？

グラフィックの美しさやキャラクターの滑らかな動きは単にプレイヤーに少々の臨場感と説得を与えるだけの「情報」にしかすぎません。その情報の載るべきゲームシステムこそ，本当に考慮されるべきではないでしょうか。

シューティングゲームのシステムにおいて，ジョイスティックとトリガボタンという体系が変わらない限り，操作性はプレイヤーの頭の中に構築された「情報」を刺激します。その結果，「情報」は単に線で連結されただけの「点と点のつながり」を超え，面としてある「世界」を構築するはずです。

そういった僕の考えをなんとかかたちにしようとしたのが今回の「ACTIVE UNIT」です。

さて，突然ですが，あなたは宇宙空間用歩兵「ACTIVE UNIT」(以下A/Uと略す）の技能試験を受けるテストパイロットです。用意された100の自動追尾ミサイルを切り抜けなくてはなりません。

なお，このゲームは操作の都合上，２トリガのジョイスティックが必要です。X1のキーボードはどうもゲームには不親切です。しかし，X1ではジョイスティックの普及率も高いでしょうから思い切ってジョイスティック専用にしました。お持ちでない方あしからず。

入力方法

このプログラムは２本のBASICプログラム（PCG定義/タイトル表示部とメインプログラムBASIC部）と１本のマシン語プログラム（マシン語サブルーチン，面データ）から構成されています。例によって，BASIC部分はX1用BASIC（CZ-8CB01，8FB01）から，マシン語部分はMACINTO-Cなどのマシン語入力ツールから入力し，間違いのないことを確認してそれぞれ，

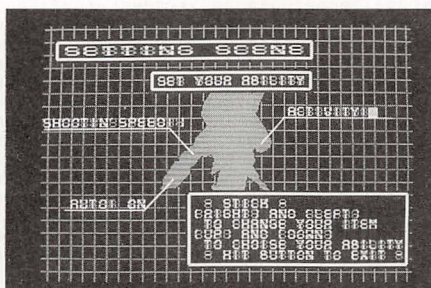
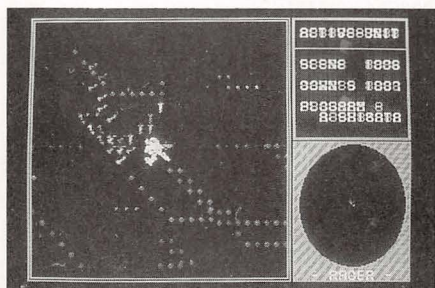
リスト１　ACTIVE.OPE
リスト２　ACTIVE.BAS
リスト３　ACTIVE.BIN

のファイル名で順番にセーブしてください（ディスクユーザーの方は順番が違ってもかまいません）。

リスト１をRUNすると初期設定が終わりしだい必要なファイルを読み込んで自動的にゲームが始まります。

ゲーム構成

プログラムの読み込みが終わるとタイトル画面が現れます。少々の堪能のあとトリガボタンを押してください。

ゲームを開始する前に使用するA/Uのアイデンティティを決定します。ジョイスティックで連射機能の有無（Auto），連射速度（Shootinspeed），敏捷性（Activity）を決めていきます。ジョイスティックの左右で項目を選択し，上下で数値を変えていってください。

連射機能はあったほうがもちろん簡単ですし，敏捷性も高いほうが機敏に動けて楽になります。皆さんの実力，上達ぐあいにあわせていろいろ変えていってください。

いよいよゲーム開始です。操作方法はちょっと込み入ってます。まず，トリガ1は弾の発射のみに使われます。スティックで8方向にバーニアの向きを変えて移動するのですが，宇宙空間なので慣性の力が大きく作用してきます。ふつうに操作しているとバーニアの向きを変えたときにA/Uの向いている方向，つまり弾を撃ち出す方向も変わってしまいますので，ここでトリガ2を使用します。トリガ2を押しながらスティックを動かすと，進行方向を変えずに弾を撃つ方向を変えることができるのです。これが実際にどういう状態なのかはゲームをやってもらうしかありません。しかし，このおかげでジョイスティック専用になったことを相殺してありあまるほど操作性は向上しました。

このようにA/Uを操作しながら迫りくるミサイルを打ち落としていくのですが，各面をクリアするにはその面に登場するミサイルの半数（小数点以下は切り上げ）を超す戦果をあげねばなりません。たとえば，3機のミサイルが出てくる面では2機以上を撃墜しなければならないわけです。

このゲームの面構成は逐次的に難しくなるようになっています。ミサイルの種類は3種，総面数100のこのゲームをあなたは連射なしで制覇できるでしょうか。

面データについて

このゲームでは，各面はミサイルの配置によって区別されています。ですからそれを記述しているデータ形式がわかれば簡単にゲームコンストラクションを楽しめるわけです。

面データはF000Hから8×3＝24バイトずつ並んでいます。1面に出てくるミサイルの最大数は8です。1バイト目がミサイルの種類（0〜2または255のみ。それ以外を指定すると暴走する），2バイト目が横座標軸，3バイト目が縦座標軸となっています。座標は－128から＋127までの2の補数で指定します。

これを元にデータを書き換えれば自分のオリジナル面のできあがりです。マシン語部分をセーブしたときの要領でセーブし，好み好みのオリジナル面も楽しんでください。

Profile

◇柴田さんは東京都にお住まいの21歳，XIDユーザーです。すでにアニメーション作品，“Rythm To Trace”やパズルゲーム“ロボット衛兵”でもお馴染みですね。

リスト1 ACTIVE .OPE

```
10 WIDTH 40:INIT:CGEN1:FOR I%=0 TO 7:PALET I%,0:NEXT
20 LOCATE 11,12:PRINT"WAIT FOR A MOMENT":SCREEN0,1,0
30 C$=HEXCHR$("3C42423C42423C00")
40 DEFCHR$(&H20)=STRING$(24,0):READ A$
50 FOR I%=&H21 TO &H5F
60 READ A$:B$=""
70 FOR J%=1 TO 8
80 B$=B$+CHR$(VAL("&H"+MID$(A$,J%*2-1,2)) OR ASC(MID$(C$,J%,1)))
90 NEXT
100 DEFCHR$(I%)=B$+HEXCHR$(A$+A$)
110 NEXT
120 FOR I%=5 TO 319 STEP 10:LINE (I%,0)-(I%,199),PSET,1,BF:NEXT
130 FOR I%=2 TO 199 STEP 10:LINE (0,I%)-(319,I%),PSET,1,BF:NEXT
140 READ X1,Y1:FOR I%=0 TO 92:READ X2,Y2
150 LINE(X1,Y1)-(X2,Y2),PSET,3:X1=X2:Y1=Y2:NEXT
160 PAINT(160,100),3,3:PAINT(160,100),1,0:RESTORE 1320
170 READ X1,Y1:FOR I%=0 TO 92:READ X2,Y2
180 LINE(X1,Y1)-(X2,Y2),PSET,0:X1=X2:Y1=Y2:NEXT
190 SCREEN0,0
200 LINE(0,0)-(215,199),PSET,4,B:LINE(2,2)-(213,197),PSET,4,B
210 SCREEN,,2:LINE(200,85)-(319,199),PSET,1,BF
220 SCREEN,,0:CIRCLE(269,142),47,4:PAINT(269,142),0,4
230 SCREEN,,3:LINE(218,94)-(319,199),PSET,1,B
240 PAINT(219,95),HEXCHR$("F5EBD7AF5FBE7DFA"),1
250 LINE(218,0)-(319,92),PSET,1,B:LINE(219,1)-(318,91),PSET,1,B
260 LINE(218,23)-(319,23),PSET,1:PSET(269,142)
270 DEFCHR$(&H60)=HEXCHR$("000000103030307183838383132200B00383C04390A3003")
280 DEFCHR$(&H61)=HEXCHR$("00000080C0E0E03C0000000080000018000000008000001C")
290 DEFCHR$(&H62)=HEXCHR$("37024C0F201C3B232A70780B3318363C33024C0F201C3B23")
300 DEFCHR$(&H63)=HEXCHR$("48F6F622C008703040F4240C8C2040E048F6F622C0087030")
310 DEFCHR$(&H64)=HEXCHR$("00061C1A040D05060E0818100101010000061C1A040D0506")
320 DEFCHR$(&H65)=HEXCHR$("0080F0B0C060A070E0E0E080C0C0C0600080F0B0C060A070")
330 DEFCHR$(&H66)=HEXCHR$("000000010307070E0000000002000000000000002000006")
340 DEFCHR$(&H67)=HEXCHR$("000000E0F2C914EA0000000080020217E8000000080020114EA")
350 DEFCHR$(&H68)=HEXCHR$("00000100000000003F2E010000000000015140D0000000000")
360 DEFCHR$(&H69)=HEXCHR$("1E24BCBE0F000604C021BD0600000400D220BCBE0F000604")
370 DEFCHR$(&H6A)=HEXCHR$("7036366C0BF33800E0E4C48F02E030F07036366C0BF33800")
380 DEFCHR$(&H6B)=HEXCHR$("0000000303000000010101030200000000000000303000000")
390 DEFCHR$(&H6C)=HEXCHR$("00007C648CD9221C70B8F040001E0C1C00007C648CD9221C")
400 DEFCHR$(&H6D)=HEXCHR$("000B1B1C6334061F050B13105A070700000B1B1C6334061F")
410 DEFCHR$(&H6E)=HEXCHR$("0818C08482800080E8D80C04020080000818C08482800080")
420 DEFCHR$(&H6F)=HEXCHR$("0000000000000010000103C0F0700000000000010140D040100")
430 DEFCHR$(&H70)=HEXCHR$("00000001030717FE0000000082C01000000000000082C010F6")
440 DEFCHR$(&H71)=HEXCHR$("8E343C1E07000604000311D0600000400082303C1E07000604")
450 DEFCHR$(&H72)=HEXCHR$("0E0400060F1E2C300001050E0E1820800200000060F1E2CB0")
460 DEFCHR$(&H73)=HEXCHR$("7036366C0BF33800E0E4C48F02E030F07036366C0BF33800")
470 DEFCHR$(&H74)=HEXCHR$("0000000000000000103070E04000000010102050000000")
480 DEFCHR$(&H75)=HEXCHR$("000B1B1C6334061F858B13105A07070080CB9B1C6334061F")
490 DEFCHR$(&H76)=HEXCHR$("00000003030000008181010302000000080C0800303000000")
500 DEFCHR$(&H77)=HEXCHR$("000000074F93285700000000648C0C81600000000648802857")
510 DEFCHR$(&H78)=HEXCHR$("00000080C0E0E06000000000000000000000000000000060")
520 DEFCHR$(&H79)=HEXCHR$("0E6C6C36D0CF1C000D6F4BC0810E180B0E6C6C36D0CF1C00")
```

```
530 DEFCHR$(&H7A)=HEXCHR$("181D3F3840A060009BD8B000800000801B1C3E3840A06000")
540 DEFCHR$(&H7B)=HEXCHR$("000000800000000FCF4000000000000A8A870800000000")
550 DEFCHR$(&H7C)=HEXCHR$("0002E59E1E6018080707051E785C00000002E59E1E601808")
560 DEFCHR$(&H7D)=HEXCHR$("000080800000000080000000000000000000080800000000")
570 DEFCHR$(&H7E)=HEXCHR$("1C18022041060100191230204201000001C18022041060100")
580 DEFCHR$(&H7F)=HEXCHR$("0030308070669C30E0E06080407830200030308070669C30")
590 DEFCHR$(&H80)=HEXCHR$("00000080C0E0987F000000000103183C000000000103187F")
600 DEFCHR$(&H81)=HEXCHR$("000000000000C04000083CF0E000000000000828B020C040")
610 DEFCHR$(&H82)=HEXCHR$("1E19026040A06000D0C0880080000801E19026040A06000")
620 DEFCHR$(&H83)=HEXCHR$("0777763BD0CF1C00076645C2810E180B0777763BD0CF1C00")
630 DEFCHR$(&H84)=HEXCHR$("B020000008B8381C0080E0F040303281A000000008B83A1D")
640 DEFCHR$(&H85)=HEXCHR$("040080800000000081010000000000000502818000000000")
650 DEFCHR$(&H86)=HEXCHR$("000000000000000080C0E070200000000004080E000000000")
660 DEFCHR$(&H87)=HEXCHR$("0430308070669C30E1E160804078302005323180706 69C30")
670 DEFCHR$(&H88)=HEXCHR$("000000000000000080C0E070200000000004080E000000000")
680 DEFCHR$(&H89)=HEXCHR$("0000030737071F73000000033218106000000000332001870")
690 DEFCHR$(&H8A)=HEXCHR$("000010F0C0FCF8C600001030301810040000103000 1C1806")
700 DEFCHR$(&H8B)=HEXCHR$("7546472106 2E2C0460747D39142C2D0174464721062E2D05")
710 DEFCHR$(&H8C)=HEXCHR$("A662C000D01C38380C4C9CCC901030302662C000D01CB8B8")
720 DEFCHR$(&H8D)=HEXCHR$("001C180C14180C000B1B130B191C0000021F1A0F14180C00")
730 DEFCHR$(&H8E)=HEXCHR$("502000202040E060C02080008060C04050A000A02040E060")
740 DEFCHR$(&H8F)=HEXCHR$("1008101818182C00101818181818183C181008101818182C00")
750 DEFCHR$(&H90)=HEXCHR$("040C001010703000040C0818107070200 40C001010703000")
760 DEFCHR$(&H91)=HEXCHR$("0003000C7830100000003060C7870700000003000C78301000")
770 DEFCHR$(&H92)=HEXCHR$("0000030278602000000000030E78E06000000000003027860 2000")
780 DEFCHR$(&H93)=HEXCHR$("0000403D7A400000000000040FFFE4000000000000403D7A400000")
790 DEFCHR$(&H94)=HEXCHR$("000020607802030000 0060E0780E030000000206078020300")
800 DEFCHR$(&H95)=HEXCHR$("001030780C000300007070780C06030000 1030780C000300")
810 DEFCHR$(&H96)=HEXCHR$("0030701010000C04207070101808 0C0400307010100 00C04")
820 DEFCHR$(&H97)=HEXCHR$("002C181818100810183C18181818181 0002C18181810 0810")
830 DEFCHR$(&H98)=HEXCHR$("000C0E0808003020040E0E08181030200 00C0E0808003020")
840 DEFCHR$(&H99)=HEXCHR$("00080C1E3000C000000E0E1E3060C00000080C1E3000C000")
850 DEFCHR$(&H9A)=HEXCHR$("000004061E40C00000000 06071E70C000000004061E40C000")
860 DEFCHR$(&H9B)=HEXCHR$("000002BC5E020000000002FF7F020000000002BC5E020000")
870 DEFCHR$(&H9C)=HEXCHR$("00C0401E0604000000C0701E0706000000C0401E06040000")
880 DEFCHR$(&H9D)=HEXCHR$("00C000301E0C080000C060301E0E0E0000C000301E0C0800")
890 DEFCHR$(&H9E)=HEXCHR$("20300008080E0C0020301018080E0E0420300008080E0C00")
900 DEFCHR$(&H9F)=HEXCHR$("003C7E6E6E7E3C00003C445C5C7C0000003C7E6E6E7E3C00")
910 DEFCHR$(&HA0)=HEXCHR$("00183E366E7C1C000018242C5C7800000 0183E366E7C1C00")
920 DEFCHR$(&HA1)=HEXCHR$("00183C766E3C18000018285C5C3800000 0183C766E3C1800")
930 DEFCHR$(&HA2)=HEXCHR$("000C3C766E3E3800000C344C5C3800000000C3C766E3E3800")
940 DEFCHR$(&HA3)=HEXCHR$("003C7E667E7E3C00003C445C5C7C0000003C7E667E7E3C00")
950 DEFCHR$(&HA4)=HEXCHR$("00303C6E767C1C0000302C567C180000000303C6E767C1C00")
960 DEFCHR$(&HA5)=HEXCHR$("00183C6E763C18000018285C5C3800000 0183C6E763C1800")
970 DEFCHR$(&HA6)=HEXCHR$("001C7C6E363E18000018685C2C380000001C7C6E363E1800")
980 DEFCHR$(&HA7)=HEXCHR$("C08000000000000000000000000000000000000000000000")
990 DEFCHR$(&HA8)=HEXCHR$("000000001818000000000000001000000000000000010000000")
1000 DEFCHR$(&HA9)=HEXCHR$("0C1E0C000000000000800000000000000000800000000000000")
1010 DEFCHR$(&HAA)=HEXCHR$("0000183C3C18000000000001000000000000000001000000000")
1020 DEFCHR$(&HAB)=HEXCHR$("0000000030F1C3800000000030F1F3F00000000030F1F39")
1030 DEFCHR$(&HAC)=HEXCHR$("00003860800000000000387CBEFEF4F80000387C96EEF4F8")
1040 DEFCHR$(&HAD)=HEXCHR$("20000100000000003F3F3F1F0F0300003E3A391F0F030000")
1050 DEFCHR$(&HAE)=HEXCHR$("00800000000000000F8E0D8BCFC98000078E058BCFC980000")
1060 DEFCHR$(&HAF)=HEXCHR$("0000061F3E78E0C00000071F3F7FFFFE0000071F3F7FF1FA")
1070 DEFCHR$(&HB0)=HEXCHR$("000000800000000001C3EAFE7F6F8FCAC1C3EA7E3F6F87C2C")
1080 DEFCHR$(&HB1)=HEXCHR$("C0400000000000000FDFAFC7A3D1E0700F5F2F07A3D1E0700")
1090 DEFCHR$(&HB2)=HEXCHR$("00001C30604000000C001C3E7569B21C0C001C3E7161B21C")
1100 DEFCHR$(&HB3)=HEXCHR$("0000061C3040808000000 71F3860E0C00000071F3860E0C0")
1110 DEFCHR$(&HB4)=HEXCHR$("0000000000000001C2281C172180C441C2281C172180C04")
1120 DEFCHR$(&HB5)=HEXCHR$("80000000000000000C1C4E460301C0600C0C0E060301C0600")
1130 DEFCHR$(&HB6)=HEXCHR$("0000182040000000A0009C2240422010000 01C224040 2010")
1140 DEFCHR$(&HBA)=HEXCHR$("0000000000000000003078783000000000000002000000000000000")
1150 RUN "ACTIVE.BAS"
1160 DATA 0000000000000000,1010100000101000,4848480000000000,24247E247E242400
1170 DATA 3C48483C12123C00,2244080010224400,3048483C48443A00,2040000000000000
1180 DATA 2040400040402000,0402020002020400,0044280028440000,0010106C10100000
1190 DATA 0000000010204000,0000003C00000000,0000000040400000,0204080010204000
1200 DATA 3C42420042423C00,0202020002020200,3C02023C40403C00,3C02023C02023C00
1210 DATA 4242423C02020200,3C40403C02023C00,3C40403C42423C00,3C02020002020200
1220 DATA 3C42423C42423C00,3C42423C02023C00,0010100010100000,0010100008102000
1230 DATA 0810200020100800,00003C003C000000,2010080008102000,3C02020C08000800
1240 DATA 0000003C42423C00,3C42423C42424200,3844483C42423C00,3C40400040403C00
1250 DATA 3844420042423C00,3C40403C40403C00,3C40403C40404000,3C40400C42423C00
1260 DATA 4242423C42424200,1010100010101000,0202020042423C00,4244483C42424200
1270 DATA 4040400040403C00,42665A0042424200,426252004A464200,1C22420042423C00
1280 DATA 3C42423C40404000,3C42420048443A00,3C42423C40444200,3C40403C02023C00
1290 DATA 7C10100010101000,4242420042423C00,4242420024241800,424242005A664200
1300 DATA 4224180018244200,8244280010101000,3E44080010227C00,3040400040403000
1310 DATA 4428107C10101000,0C02020002020C00,1028440000000000,0000000000003C00
1320 DATA 138, 78,134, 72,133, 69,135, 59,142, 51,152, 46,161, 45,173, 49
1330 DATA 177, 53,181, 48,183, 48,188, 51,188, 53,185, 58,182, 67,180, 68
1340 DATA 190, 82,194, 91,193, 95,185,100,181, 99,179, 96,178, 98,180,106
1350 DATA 178,110,184,118,183,119,186,122,187,122,193,129,191,131,190,131
1360 DATA 183,126,183,125,179,122,178,122,173,118,171,128,168,130,170,131
1370 DATA 172,135,171,138,175,141,177,149,173,152,174,155,176,156,178,160
1380 DATA 148,160,134,160,136,156,144,153,146,152,147,150,149,149,151,139
1390 DATA 147,138,142,126,143,123,146,124,149,113,143,117,141,116,143,108
1400 DATA 144,104,142,101,137,103,134,102,132,104,131,106,129,108,127,110
1410 DATA 126,110,125,112,125,115,114,126,111,127,107,127,102,129, 99,127
1420 DATA 101,123,101,120,112,105,112,105,115,105,115,104,114,103,120, 97
1430 DATA 122, 97,124, 94,127, 92,128, 88,133, 83,138, 78
```

リスト2 ACTIVE .BAS

```
10 CLEAR &HDFFF
20 IF MEM$(&HE000,3)<>HEXCHR$("CD9AE9")THEN LOADM"ACTIVE.BIN"
30 GOSUB"OPEN":GOSUB"MAKING"
40 ' INIT
50 MEM$(&HEBEF,9)=HEXCHR$("000000100000000000")
60 POKE &HECB9,0:POKE &HECB4,0,&HF0:POKE &HE14E,0
70 SCREEN1,0:RESTORE 1410:GOSUB 1380:GOSUB"SCREEN":SCREEN 0
80 SC%=0:SD%=1:AP%=1:PLAY750:CALL &HE0A5:PLAY"F3:A3:+C3":CALL &HE000
90 ' MAIN
100 IF PEEK(&HECB9)=1 THEN "OVER" ELSE  GOSUB "INFORM"
110 IF SC%=100 THEN "CLEAR"
120 CALL &HE003
130 IF PEEK(&HECB9)=0 GOSUB "INFORM":GOTO 110 ELSE "OVER"
140 ' INFORMATION
150 LABEL"INFORM"
160 SD%=SD%+PEEK(&HECB7)-PEEK(&HECB8):AP%=AP%+PEEK(&HECB7):SC%=SC%+1
170 LOCATE 36,4:PRINT RIGHT$("00"+RIGHT$(STR$(SC%),LEN(STR$(SC%))-1),3)
180 LOCATE 36,6:PRINT RIGHT$("00"+RIGHT$(STR$(SD%-1),LEN(STR$(SD%-1))-1),3)
190 RETURN
200 LABEL"CLEAR"
210 SCREEN 0,1:GOSUB"PALET":CLS:CLS3:CLS2:SCREEN 1
220 X1=22:Y1=70:X2=298:Y2=137:GOSUB"WINDOW"
230 RESTORE 260:FOR Y%=9 TO 16
240 READ M$:GOSUB"MPRINT":NEXT
250 IF STRIG(1)=0 THEN 250 ELSE 410
260 DATA"CONGRATULATIONS !","YOUR ABILITY HAS REACHED THE POINT"
270 DATA"ENOUGH TO USE THIS UNIT.","HAVING SOME ROOM,","YOU SHOULD PLAY","IN MOR
E HARD SITUATION.","","PUSH TO OUIT"
280 ' GAME OVER
290 LABEL"OVER":SOUND 7,255
300 FOR I%=0 TO 100:CALL &HE651:PLAY"R0":NEXT
310 FOR I%=1 TO 13:PLAY"C1:-B1:E1":FOR J%=0 TO 2
320 LOCATE 7,12:PRINT LEFT$("GAME IS OVER",I%);
330 CALL &HE651:NEXT:NEXT
340 FOR I%=0 TO 100:CALL &HE651:LOCATE 7,12:PRINT"GAME IS OVER":NEXT
350 FOR I%=1 TO 13:PLAY"C1:-B1:E1":FOR J%=0 TO 1
360 LOCATE 7,22:PRINT LEFT$("PUSH TO QUIT",I%);
370 LOCATE 7,12:PRINT"GAME IS OVER"
380 CALL &HE651:NEXT:NEXT
390 LOCATE 7,12:PRINT"GAME IS OVER":LOCATE 7,22:PRINT "PUSH TO QUIT"
400 CALL &HE651:IF STRIG(1)=0 THEN 390
410 SCREEN 0,1:GOSUB"PALET":CLS:CLS3:CLS2:SCREEN 1
420 X1=45:Y1=62:X2=274:Y2=128:GOSUB"WINDOW"
430 Y%=8:M$="- RESULTS -":GOSUB"MPRINT"
440 Y%=10:M$="NOUMBER OF APPEARNCE :"+RIGHT$("  "+STR$(AP%-1),4):GOSUB"MPRINT"
450 Y%=11:M$="SHOOTING DOWN        :"+RIGHT$("  "+STR$(SD%-1),4):GOSUB"MPRINT"
460 Y%=12:M$="SHOOTING RETIO       :"+STR$(INT(SD%/AP%*100))+"%":GOSUB"MPRINT"
470 P%=INT(100+(AU=0)*20-(50-(SD%/AP%*100)/2)-(AC<2)*10-(100-SC%))
480 Y%=13:M$="YOUR TOTAL POINT     :"+RIGHT$("  "+STR$(P%),4):GOSUB"MPRINT"
490 Y%=15:M$="PUSH TO QUIT":GOSUB"MPRINT"
500 IF STRIG(1)=0 THEN 500 ELSE GOTO 30
510 ' SETTING OF PLAYER'S ABILITY
520 LABEL"MAKING"
530 CLS2:CLS3:CSIZE:CLS
540 X1=11:Y1=13:X2=229:Y2=25:GOSUB"WINDOW"
550 LOCATE 2,2:CSIZE2:PRINT#0,"SETTING SCENE"
560 X1=124:Y1=132:X2=314:Y2=186:GOSUB"WINDOW"
570 LOCATE 17,17:PRINT"- STICK -"
580 PRINT TAB(16);"(RIGHT) AND (LEFT)"
590 PRINT TAB(17);"TO CHANGE YOUR ITEM"
600 PRINT TAB(16);"(UP) AND (DOWN)"
610 PRINT TAB(17);"TO CHOISE YOUR ABILITY"
620 PRINT TAB(17);"- HIT BUTTON TO EXIT -"
630 SCREEN,,3:LINE(110,120)-(86,144)-(20,144)
640 LINE(127,100)-(107,80)-(2,80)
650 LINE(185,92)-(205,72)-(300,72)
660 LOCATE 0,9:PRINT"SHOOTIN'SPEED:":LOCATE 3,17:PRINT"AUTO:"
670 LOCATE 26,8:PRINT"ACTIVITY:"
680 X1=92:Y1=37:X2=228:Y2=49:GOSUB"WINDOW"
690 M$="SET YOUR ABILITY":Y%=5:GOSUB"MPRINT"
700 PLAY500:S=6:GOSUB 850:GOSUB 960:ITEM=1
710 PLAY"C1R4:-B1R4:E1R4":ON ITEM+1 GOSUB 830,890,940
720 IF STRIG(1)<>-1 GOTO 710
730 IF AU=0 THEN POKE &HEB67,1 ELSE POKE &HEB67,0
740 POKE &HEB65,5-AC:POKE &HEB66,6-SS*2
750 CLS2:CLS3:CFLASH:CSIZE:CLS:X1=53:Y1=53:X2=266:Y2=143:GOSUB"WINDOW"
760 Y%=7:M$="YOUR ABILITY":GOSUB"MPRINT"
770 Y%=10:M$="ACTIVITY        :"+STR$(AC+1):GOSUB"MPRINT"
780 Y%=12:M$="AUTO REPEATING :"
790 IF AU=0 THEN M$=M$+" ON" ELSE M$=M$+"OFF":GOSUB"MPRINT":GOTO 810
800 GOSUB"MPRINT":Y%=14:M$="SHOOTING SPEED :"+STR$(SS+1):GOSUB"MPRINT"
810 Y%=17:M$="- HIT BUTTON TO EXIT -":GOSUB"MPRINT"
820 IF STRIG(1)<>-1 THEN 820 ELSE RETURN
830 LOCATE 14,9:CFLASH1:PRINTUSING"#";SS+1
840 S=STICK(1):IF STRIG(1) THEN RETURN ELSE IF S=0 THEN 840
850 IF S=6 OR S=4 THEN ITEM=(ITEM-(S=6)+(S=4)) MOD 3:ITEM=ITEM-(ITEM=-1)*3:LOCAT
E 14,9:CFLASH:PRINTUSING"#";SS+1:RETURN
860 IF S=2 AND SS<>0 THEN SS=SS-1
870 IF S=8 AND SS<>2 THEN SS=SS+1
880 PAUSE2:GOTO 830
890 LOCATE 8,17:CFLASH1:IF AU=0 THEN PRINT" ON" ELSE PRINT"OFF"
```

▶私は二十世紀梨ドリンクが好きである。文句あっか。　横田　紀明（22）山口県

```
900 S=STICK(1):IF STRIG(1) THEN RETURN ELSE IF S=0 THEN 900
910 IF S=4 OR S=6 THEN ITEM=(ITEM-(S=6)+(S=4)) MOD 3:ITEM=ITEM-(ITEM=-1)*3:LOCAT
E 8,17:CFLASH:PRINTSCRN$(8,17,3):RETURN
920 IF S=2 THEN AU=0 ELSE IF S=8 THEN AU=1
930 PAUSE2:GOTO 890
940 LOCATE 35,8:CFLASH1:PRINTUSING"#";AC+1
950 S=STICK(1):IF STRIG(1) THEN RETURN ELSE IF S=0 THEN 950
960 IF S=4 OR S=6 THEN ITEM=(ITEM-(S=6)+(S=4)) MOD 3:ITEM=ITEM-(ITEM=-1)*3:LOCAT
E 35,8:CFLASH:PRINTUSING"#";AC+1:RETURN
970 IF S=2 AND AC<>0 THEN AC=AC-1
980 IF S=8 AND AC<>4 THEN AC=AC+1
990 PAUSE2:GOTO 940
1000 ' MAIN SCREEN INIT
1010 LABEL"SCREEN":CSIZE0:CLS
1020 LOCATE 28,1:PRINT"ACTIVE-UNIT"
1030 LOCATE 28,4:PRINT"SCENE  :000"
1040 LOCATE 28,6:PRINT"DOWN'S :000"
1050 LOCATE 28,8:PRINT"PLOGRAM -"
1060 LOCATE 28,9:PRINT"  A.SHIBATA"
1070 LOCATE 29,24:COLOR5:PRINT"- RADER -";
1080 COLOR7:FOR I%=0 TO 24:LOCATE 0,I%:PRINT STRING$(25," ");:NEXT:RETURN
1090 ' OPENING
1100 LABEL"OPEN"
1110 INIT:SCREEN1,1,3:GOSUB"PALET0":CLS2:CLS3:CGEN1:CLS
1120 X1=11:Y1=13:X2=197:Y2=25:GOSUB"WINDOW"
1130 X1=196:Y1=142:X2=307:Y2=169:GOSUB"WINDOW"
1140 RESTORE 1200:FOR I%=0 TO 6
1150 READ X,Y,L:GOSUB"INFOR":NEXT:GOSUB"PALET"
1160 LOCATE 2,2:CSIZE 2:COLOR 2:PRINT #0,"ACTIVE UNIT"
1170 LOCATE 25,18:COLOR 7:PRINT "COPYRIGHT":PRINT TAB(28);"A.SHIBATA"
1180 PRINT TAB(33);"1989"
1190 GOTO"WAIT TRIG"
1200 DATA 155,130,5,110,120,8,140, 90,4,150, 55,3,185, 90,9,183, 52,6,173,105,7
1210 ' WINDOW OPEN
1220 LABEL"WINDOW"
1230 SCREEN,,2:LINE(X1-3,Y1-3)-(X2+3,Y2+3),PSET,1,BF:SCREEN,,3
1240 LINE(X1,Y1-1)-(X2,Y1),PSET,1,BF:LINE(X1,Y2)-(X2,Y2+1),PSET,1,BF
1250 LINE(X1-1,Y1)-(X1,Y2),PSET,1,BF:LINE(X2,Y1)-(X2+1,Y2),PSET,1,BF
1260 RETURN
1270 ' LINE OF POINTING
1280 LABEL"INFOR":SCREEN,,3
1290 A=-1:IF X>160 THEN A=1
1300 B=-1:IF Y>100 THEN B=1
1310 LINE (X,Y)-(X+A*30,Y+B*15),PSET,1
1320 LINE-(X+A*(34+L*3),Y+B*15),PSET,1
1330 LINE (X+A*(33+L*3),Y+B*15-3)-(X+A*33,Y+B*15-2),PSET,BF,HEXCHR$("DAD6")
1340 RETURN
1350 ' PALET INITIALIZE
1360 LABEL"PALET"
1370 RESTORE 1400
1380 FOR I%=1 TO 7:READ A%
1390 PALET I%,A%:NEXT:RETURN
1400 DATA 1,0,0,7,7,7,7
1410 DATA 2,0,0,1,1,1,1
1420 LABEL"PALET0"
1430 FOR I%=1 TO 7:PALET I%,0:NEXT:RETURN
1440 ' WAIT FOR PUSHING
1450 LABEL"WAIT TRIG"
1460 M$="PUSH TO QUIT":Y%=22:GOSUB"MPRINT"
1470 IF STRIG(1) THEN RETURN ELSE 1470
1480 ' MESSAGE PRINT
1490 LABEL"MPRINT"
1500 LOCATE 20-LEN(M$)¥2,Y%:PLAY 3000
1510 FOR I%=1 TO LEN(M$):PRINT MID$(M$,I%,1);
1520 PLAY"G3R:A3R:C3R":NEXT:RETURN
```

リスト3 ACTIVE .BIN

```
E000 CD 9A E9 CD E3 E9 CD B3 : 69
E008 E2 CD 4F E1 CD 51 E6 CD : B0
E010 A5 E0 CD 9A E1 CD 43 E3 : C0
E018 CD 42 E7 0E 1E CD EF E7 : C5
E020 3A B6 EC FE 00 CA 2B E0 : AF
E028 C3 06 E0 2A B7 EC E5 06 : 61
E030 64 C5 CD B3 E2 CD 4F E1 : 88
E038 CD 51 E6 CD A5 E0 CD 9A : BD
E040 E1 CD 42 E7 0E 1E CD EF : BF
E048 E7 C1 10 E5 E1 7C B7 CB : 7C
E050 1D BD CA 80 E0 FA 80 E0 : 5E
E058 3E 01 32 B9 EC 21 81 E0 : 98
E060 11 F8 EB 01 24 00 ED B0 : B6
E068 06 64 C5 CD 51 E6 CD 43 : 43
E070 E3 CD 4F E1 CD 42 E7 0E : E4
E078 23 CD EF E7 C1 10 EB C9 : 4B
---------------------------------
SUM: 8F 9D A7 99 AB 24 22 EF A1DB

E080 C9 03 01 00 00 00 00 00 : CD
E088 00 00 03 FE 00 00 00 00 : 01
E090 00 00 00 03 00 01 00 00 : 04
E098 00 00 00 00 03 00 FF 00 : 02
E0A0 00 00 00 00 00 3E 00 32 : 70
E0A8 61 EB 32 62 EB 3A 59 EB : 49
E0B0 FE 00 CA 14 E1 47 32 5F : 95
E0B8 EB 3A 5B EB FE 01 CA 14 : 48
E0C0 E1 11 0C 08 CD 81 E7 3E : 79
E0C8 00 32 60 EB 3A 4E E1 3C : 22
E0D0 32 4E E1 4F 3A 65 EB B9 : F3
E0D8 C2 1A E1 3E 00 32 4E E1 : 5C
E0E0 78 21 3E E1 3D 07 85 6F : F0
E0E8 46 23 4E 21 F3 EB 7E 80 : B4
E0F0 FE 09 C2 F7 E0 3E 08 FE : E4
E0F8 F7 C2 FE E0 3E F8 77 23 : 67
---------------------------------
SUM: 9B E2 D5 BB 5C 4F D7 B4 72F2

E100 7E 81 FE 09 C2 09 E1 3E : F0
E108 08 FE F7 C2 10 E1 3E F8 : E6
E110 77 C3 1A E1 11 00 08 CD : 1B
E118 81 E7 21 EF EB 46 23 4E : 1A
E120 2B CD 31 E8 78 96 32 61 : B2
E128 EB 23 79 96 32 62 EB 60 : FC
E130 69 3A 5F EB 3D 4F 3A 60 : 13
E138 EB 81 CD 04 E7 C9 FF 01 : ED
E140 00 01 01 01 FF 00 01 00 : 03
E148 FF FF 00 FF 01 FF 00 3A : 37
E150 5A EB FE 01 C0 21 64 EC : 75
E158 06 0A CD 73 E2 D8 3A 5F : A3
E160 EB 3D E5 D9 21 82 E1 47 : B1
E168 80 80 85 6F 01 03 00 D1 : C9
E170 ED B0 3E 10 12 D9 3E 08 : 1C
E178 32 60 EB 21 80 02 22 2C : 6E
---------------------------------
SUM: D1 96 65 F5 F2 98 80 44 5610

E180 E2 C9 0B 0D 0A 0C 0D 08 : EE
E188 0E 0D 06 0B 0C 0C 0E 0C : 5E
E190 04 0B 0B 0E 0C 0B 00 0E : 4D
E198 0B 02 ED 4B 61 EB 11 00 : A2
E1A0 00 ED 53 61 EB C5 21 64 : D6
E1A8 EC 06 0A 11 07 00 7E 23 : B5
E1B0 FE FF CA FA E1 32 63 EB : 22
E1B8 7E 32 64 EB D9 21 55 EB : 39
E1C0 CD 4A E7 D9 2B CD CC E8 : 83
E1C8 7E 32 63 EB 23 7E 32 64 : 35
E1D0 EB CD 2E E2 D2 E1 E1 CD : 29
E1D8 85 E2 2B 36 FF 23 C3 FA : A7
E1E0 E1 3E 00 32 80 E7 D9 21 : B2
E1E8 2B E2 CD 4A E7 D9 3A 80 : 9E
E1F0 E7 FE 00 C2 FA E1 2B 36 : E3
E1F8 FF 23 19 10 B1 C1 ED 43 : ED
---------------------------------
SUM: 14 73 1D F2 60 D7 50 AC CE80
```

▶女の子と歩いていると，どこからかベートーベンの月光がかかった（偶然にも私はテグザーのおかげで覚えていたのだ)。かっこつけるために「あ，ムーンライトソナタだね」とかほざいたら，彼女曰く「あら，そうね，第○楽章のあたりね」などと返され墓穴を掘ってしまった私は（て）氏に一歩近づけたような気がします。　　下田　達也（22）三重県

```
E200 61 EB ED 5B 2C E2 3E C0 : A0
E208 BB CA 24 E2 CD 81 E7 7B : 3B
E210 C6 08 5F ED 53 2C E2 11 : 8C
E218 10 09 CD 81 E7 11 00 0D : 6C
E220 CD 81 E7 C9 11 00 09 CD : E5
E228 81 E7 C9 BA C0 00 C5 D5 : 45
E230 E5 3A 63 EB D6 0C 47 3A : D0
E238 64 EB D6 0C 4F 21 F8 EB : 84
E240 11 09 00 D9 06 08 D9 7E : 58
E248 FE FF CA 6A E2 FE 03 CA : DE
E250 6A E2 23 7E 23 B8 C2 68 : F2
E258 E2 7E B9 C2 68 E2 D9 78 : 76
E260 32 B2 E2 D9 37 C3 6F E2 : EA
E268 2B 2B 19 D9 10 D8 D9 E1 : EA
E270 D1 C1 C9 C5 D5 11 08 00 : 0E
E278 7E FE FF CA 82 E2 19 10 : D2
-----------------------------------
SUM: 90 57 8F E9 3A FB F4 1B C26D

E280 F7 37 D1 C1 C9 C5 D5 E5 : 08
E288 3A B2 E2 47 3E 08 90 21 : 0C
E290 F8 EB 11 09 00 CA 9C E2 : 45
E298 47 19 10 FD 36 03 23 23 : EC
E2A0 23 36 00 21 80 04 22 C1 : E1
E2A8 E4 3E 1F 32 C3 E4 E1 D1 : CC
E2B0 C1 C9 08 C5 D5 E5 3E 00 : 4F
E2B8 32 59 EB 32 5A EB 32 5B : 7A
E2C0 EB 3A 67 EB FE 01 C2 E5 : 1D
E2C8 E2 3A 42 E3 3C 32 42 E3 : D4
E2D0 47 3A 66 EB B8 C2 E5 E2 : 13
E2D8 3E 00 32 42 E3 3A 5C EB : 16
E2E0 F6 BF 32 5C EB 3E 0E 01 : 7B
E2E8 00 1C ED 79 05 ED 78 FE : EA
E2F0 FF CA 33 E3 57 E6 0F 21 : 4C
E2F8 3A E3 06 08 BE CA 03 E3 : 99
-----------------------------------
SUM: EB B9 7F 13 89 5C 74 90 4241

E300 23 10 F9 78 32 59 EB 1E : 38
E308 00 3A 5C EB CB 6F CA 18 : 9D
E310 E3 CB 6A C2 18 E3 1E 01 : F4
E318 7B 32 5A EB CB 72 C2 2D : 1E
E320 E3 3E 01 32 5B EB 7A 32 : 46
E328 5C EB C3 36 E3 3E 00 32 : 93
E330 5B EB 7A 32 5C EB E1 D1 : EB
E338 C1 C9 06 0E 0A 07 0B 05 : BF
E340 0D 09 00 21 F8 EB 06 08 : 28
E348 7E FE FF CA 06 E4 23 7E : D0
E350 C6 0C 32 63 EB 23 7E C6 : B9
E358 0C 32 64 EB D9 21 55 EB : C7
E360 CD 4A E7 3A 63 EB CB 2F : 80
E368 C6 80 06 00 4F 21 87 00 : 43
E370 09 44 4D 3A 64 EB CB 2F : 1D
E378 C6 87 16 00 5F 3E 00 08 : 08
-----------------------------------
SUM: 9B FE 42 65 BB 80 14 3B AD2E

E380 CD 90 E7 D9 2B 2B 7E D9 : CA
E388 4F D9 23 11 0F E4 CB 27 : 41
E390 83 5F ED 53 98 E3 ED 5B : E5
E398 15 E4 ED 53 9F E3 CD 61 : E9
E3A0 E4 2B 7E FE 03 CA 06 E4 : 42
E3A8 FE FF CA 06 E4 23 7E C6 : 18
E3B0 0C 32 63 EB 23 7E C6 0C : FF
E3B8 32 64 EB 23 CD 09 E6 D2 : 32
E3C0 CF E3 2B 2B 2B 3A B8 EC : 11
E3C8 3C 32 B8 EC C3 06 E4 7E : 3D
E3D0 D9 21 14 EB 85 CB 81 CB : 95
E3D8 21 CB 21 CB 21 81 6F CD : B6
E3E0 4A E7 3A 63 EB CB 2F C6 : 79
E3E8 80 06 00 4F 21 87 00 09 : 86
E3F0 44 4D 3A 64 EB CB 2F C6 : DA
E3F8 87 16 00 5F 3E FF 08 CD : 0E
-----------------------------------
SUM: 6E BD 06 E4 11 F1 25 A8 28CA

E400 90 E7 D9 2B 2B 2B 11 09 : EB
E408 00 19 05 C2 48 E3 C9 17 : EB
E410 E4 3C E4 17 E4 61 E4 C5 : 09
E418 D5 E5 11 03 E6 ED A0 ED : 2E
E420 A0 ED A0 2B CD C4 E4 3A : 07
E428 58 EB E6 07 C2 33 E4 3A : 43
E430 05 E6 77 2B 2B CD CC E8 : 39
E438 E1 D1 C1 C9 C5 D5 E5 11 : CC
E440 03 E6 ED A0 ED A0 ED A0 : 90
E448 2B CD C4 E4 3A 58 EB E6 : 03
E450 0F CA 58 E4 3A 05 E6 77 : B1
E458 2B 2B CD CC E8 E1 D1 C1 : 4A
E460 C9 C5 D5 E5 7E C6 0C 32 : CA
E468 63 EB 23 7E C6 0C 32 64 : 57
E470 EB 23 7E CB 2F CB 2F CD : 4D
E478 C9 E6 7E 3C 77 47 78 FE : 9D
-----------------------------------
SUM: 6F 11 5B CB EF B7 4B 5E F54E

E480 0D C2 99 E4 2B 2B 2B 36 : 03
E488 FF 11 00 0A CD 81 E7 3A : 89
E490 B6 EC 3D 32 B6 EC C3 BD : 33
E498 E4 2A C1 E4 11 C0 00 19 : 9D
E4A0 16 04 5D CD 81 E7 14 5C : 1C
E4A8 CD 81 E7 22 C1 E4 3A C3 : F9
E4B0 E4 3D 32 C3 E4 CB 2F 16 : 0A
E4B8 0A 5F CD 81 E7 E1 D1 C1 : 11
E4C0 C9 80 31 E3 C5 D5 E5 3A : 16
E4C8 03 E6 ED 44 32 03 E6 3A : 6F
E4D0 04 E6 ED 44 32 04 E6 3A : 71
E4D8 03 E6 CB 7F C2 19 E5 3A : 2D
E4E0 03 E6 32 06 E6 3A 04 E6 : 2B
E4E8 CB 7F CA 05 E5 3A 04 E6 : 22
E4F0 ED 44 32 07 E6 CD 96 E5 : 98
E4F8 3A 08 E6 47 3E 04 90 32 : 73
-----------------------------------
SUM: 3F ED C4 7A A6 09 E7 07 F198

E500 08 E6 C3 52 E5 3A 04 E6 : 0C
E508 32 07 E6 CD 96 E5 3A 08 : A9
E510 E6 C6 04 32 08 E6 C3 52 : E5
E518 E5 3A 03 E6 ED 44 32 06 : 71
E520 E6 3A 04 E6 CB 7F C2 3F : 55
E528 E5 3A 04 E6 32 07 E6 CD : F5
E530 96 E5 3A 08 E6 47 3E 0C : 34
E538 90 32 08 E6 C3 52 E5 3A : E4
E540 04 E6 ED 44 32 07 E6 CD : 07
E548 96 E5 3A 08 E6 C6 0C 32 : A7
E550 08 E6 3A 05 E6 47 3A 08 : 9C
E558 E6 B8 FA 78 E5 CA 8F E5 : 33
E560 90 FE 08 FA 6F E5 3A 05 : 23
E568 E6 3D E6 0F C3 8F E5 3A : 89
E570 05 E6 3C E6 0F C3 8F E5 : 53
E578 4F 78 91 FE 08 FA 89 E5 : C6
-----------------------------------
SUM: 48 7A 10 A7 42 77 F0 8D 045A

E580 3A 05 E6 3C E6 0F C3 8F : A8
E588 E5 3A 05 E6 3D E6 0F 32 : 6E
E590 05 E6 E1 D1 C1 C9 3A 06 : 67
E598 E6 47 3A 07 E6 B8 F2 A5 : A3
E5A0 E5 CD C1 E5 C9 3A 07 E6 : 48
E5A8 47 3A 06 E6 32 07 E6 78 : 04
E5B0 32 06 E6 CD C1 E5 3A 08 : D3
E5B8 E6 47 3E 04 90 32 08 E6 : 1F
E5C0 C9 26 00 3A 07 E6 6F 44 : C9
E5C8 4D 29 29 09 06 00 3A 06 : EE
E5D0 E6 4F B7 ED 42 D2 DD E5 : AF
E5D8 16 00 C3 FE E5 26 00 3A : 1C
E5E0 07 E6 6F 44 4D 29 09 06 : 25
E5E8 00 3A 06 E6 4F CB 21 CB : 2C
E5F0 10 B7 ED 42 C2 FC E5 16 : AF
E5F8 01 C3 FE E5 16 02 7A 32 : 6B
-----------------------------------
SUM: 78 F8 F4 15 BE 9E 3C 3A 2EA5

E600 08 E6 C9 FE 00 0C 02 00 : C3
E608 0C 3A 63 EB D6 1B D0 3A : 8F
E610 64 EB D6 18 D0 E5 D5 C5 : 8C
E618 26 00 3A 64 EB 6F 54 5D : CF
E620 29 29 19 29 29 29 16 00 : FC
E628 3A 63 EB 5F 19 11 00 30 : 41
E630 19 44 4D ED 78 FE 60 FA : 67
E638 4C E6 FE 8F F2 4C E6 C1 : A4
E640 C5 78 32 B2 E2 CD 85 E2 : 37
E648 37 C3 4D E6 B7 C1 D1 E1 : 57
E650 C9 E5 C5 F5 D5 21 68 EB : B1
E658 06 0F 3A F3 EB ED 44 57 : B5
E660 3A F4 EB ED 44 5F 7E 32 : 59
E668 63 EB 23 7E 32 64 EB D9 : 49
E670 21 55 EB CD 4A E7 D9 23 : 5B
E678 23 23 72 23 73 2B 2B 2B : CF
-----------------------------------
SUM: 12 47 74 44 C9 70 C6 A5 E432

E680 2B 2B CD 31 E8 CD 31 E8 : 22
E688 7E E6 1F 77 32 63 EB 23 : 9D
E690 7E E6 1F 77 32 64 EB 23 : 9E
E698 23 7E D9 21 44 EB 85 6F : BE
E6A0 CD 4A E7 D9 D5 11 06 00 : C3
E6A8 19 D1 10 BA D1 F1 C1 E1 : 18
E6B0 00 00 00 00 00 00 00 00 : 00
E6B8 00 00 00 00 00 00 00 00 : 00
E6C0 00 00 00 00 00 00 00 00 : 00
E6C8 C9 C5 D5 E5 21 34 EB CB : 53
E6D0 27 CB 27 85 6F CD 4A E7 : 0B
E6D8 3A 63 EB 3C 32 63 EB 23 : 67
E6E0 CD 4A E7 3A 64 EB 3C 32 : F5
E6E8 64 EB 3A 63 EB 3D 32 63 : A9
E6F0 EB 23 CD 4A E7 3A 63 EB : 94
E6F8 3C 32 63 EB 23 CD 4A E7 : DD
-----------------------------------
SUM: B2 0D 13 4B 51 14 8E BA C2A0

E700 E1 D1 C1 C9 F5 C5 D5 E5 : B0
E708 21 54 EA 57 87 82 07 07 : CD
E710 06 00 4F 09 3E 0B 32 63 : 3C
E718 EB 32 64 EB CD 4A E7 ED : 57
E720 4B 7F E7 11 03 04 7E ED : 34
E728 79 23 03 15 C2 26 E7 E5 : 68
E730 21 24 00 09 44 4D E1 16 : D6
E738 04 1D C2 26 E7 E1 D1 C1 : 63
E740 F1 C9 3A 58 EB 3C 32 58 : FD
E748 EB C9 3A 63 EB D6 1B D0 : FD
E750 3A 64 EB D6 18 D0 C5 D5 : E1
E758 E5 26 00 3A 64 EB 6F 54 : 57
E760 5D 29 29 19 29 29 29 16 : 59
E768 00 3A 63 EB 5F 19 11 00 : 11
E770 30 19 44 4D E1 7E ED 79 : 9F
E778 ED 43 7F E7 D1 C1 C9 2B : 1C
-----------------------------------
SUM: 51 15 B8 6C 03 42 7D F0 C5C7

E780 31 C5 D5 E5 01 00 1C ED : BA
E788 51 05 ED 59 E1 D1 C1 C9 : D8
E790 C5 D5 E5 21 D2 00 B7 ED : 16
E798 42 D2 E3 E7 21 3D 01 B7 : F4
E7A0 ED 42 DA E3 E7 3E F8 A3 : AC
E7A8 62 6F 3E 07 A3 07 07 07 : CE
E7B0 B2 F6 40 57 5D 29 29 19 : 07
E7B8 3E F8 51 A1 4F B7 CB 18 : 11
E7C0 CB 19 CB 18 CB 19 CB 18 : 8E
E7C8 CB 19 09 44 4D 21 E7 E7 : 6D
E7D0 7A E6 07 85 6F 56 ED 78 : 16
E7D8 A2 5F 7A EE FF 57 08 A2 : 69
E7E0 B3 ED 79 E1 D1 C1 C9 7F : D4
E7E8 BF DF EF F7 FB FD FE D9 : 53
E7F0 06 32 ED 44 ED 44 ED 44 : CB
E7F8 ED 44 10 F6 D9 0D C2 EF : CE
-----------------------------------
SUM: DF C9 ED 09 23 29 A5 D9 F2B1

E800 E7 C9 C5 D5 E5 21 2F E8 : 67
E808 CD 27 E8 F5 D1 21 30 E8 : DB
E810 CD 27 E8 F5 C1 79 AB 1F : D5
E818 2A 56 EB ED 6A 22 56 EB : 25
E820 E1 D1 C1 3A 56 EB C9 46 : FD
E828 2A 56 EB 29 10 FD C9 10 : 7A
E830 02 F5 C5 D5 E5 DD E1 DD : 11
E838 7E 03 DD 86 06 CB 67 C2 : DE
E840 48 E8 DD 77 06 C3 C5 E8 : FA
E848 D6 10 DD 77 06 DD 7E 04 : 9F
E850 DD 86 07 CB 7F C2 72 E8 : D0
E858 CB 67 C2 63 E8 DD 77 07 : 9A
E860 C3 89 E8 D6 10 DD 77 07 : 75
E868 DD 7E 00 3C DD 77 00 C3 : AE
E870 89 E8 CB 67 CA 7D E8 DD : AF
E878 77 07 C3 89 E8 C6 10 DD : 65
-----------------------------------
SUM: 9C 67 C7 88 44 43 D5 2E A767

E880 77 07 DD 7E 00 3D DD 77 : 6A
E888 00 DD 7E 05 DD 86 08 CB : 96
E890 7F C2 AE E8 CB 67 C2 9F : 6A
E898 E8 DD 77 08 C3 C5 E8 D6 : 8A
E8A0 10 DD 77 08 DD 7E 01 3C : 04
E8A8 DD 77 01 C3 C5 E8 CB 67 : F7
E8B0 CA B9 E8 DD 77 08 C3 C5 : 4F
E8B8 E8 C6 10 DD 77 08 DD 7E : 75
E8C0 01 3D DD 77 01 DD E5 E1 : 36
E8C8 D1 C1 F1 C9 F5 C5 D5 E5 : C0
E8D0 DD E1 DD 7E 03 DD 86 06 : 85
E8D8 CB 67 C2 E3 E8 DD 77 06 : 19
E8E0 C3 50 E9 D6 10 DD 77 06 : 3C
E8E8 DD 7E 02 E6 07 21 92 E9 : E6
E8F0 85 6F DD 7E 04 86 FE 02 : D9
E8F8 F2 01 E9 DD 77 04 C3 19 : 10
-----------------------------------
SUM: 0E DA 0E B0 6E 49 7C 79 FE13

E900 E9 3E 00 DD 77 04 DD 46 : A2
E908 00 DD 7E 02 FE 08 F2 15 : 6A
E910 E9 04 C3 16 E9 05 DD 70 : 01
E918 00 DD 7E 02 D6 04 E6 07 : 24
E920 21 92 E9 85 6F DD 7E 05 : F0
E928 86 FE 02 F2 34 E9 DD 77 : E9
E930 05 C3 50 E9 3E 00 DD 77 : 93
E938 05 DD 46 01 DD 7E 02 C6 : 4C
E940 04 E6 0F FE 08 F2 4C E9 : 26
E948 05 C3 4D E9 04 DD 70 01 : 50
E950 3A 61 EB DD 86 00 DD 77 : 3D
E958 00 3A 62 EB DD 86 01 DD : C8
E960 77 01 D1 C1 F1 DD E5 E1 : 9E
E968 C9 E5 D5 C5 78 D9 47 D9 : B9
E970 06 00 C5 D9 D9 C1 C5 21 : 24
E978 85 E9 ED B0 D9 10 F5 D9 : C2
-----------------------------------
SUM: 91 3F 41 16 7C 35 4C 7D 69D1

E980 C1 C1 D1 E1 C9 FF FF FE : F9
E988 00 00 00 00 00 00 00 00 : 00
E990 00 00 00 01 02 02 02 02 : 09
E998 02 01 21 68 EB 06 0F 11 : 9D
E9A0 06 00 CD 02 E8 E6 1F 77 : 39
E9A8 23 CD 02 E8 E6 1F 77 3E : 94
E9B0 00 23 77 CD 02 E8 E6 03 : 3A
E9B8 3C D9 47 3E 01 07 10 FD : AF
E9C0 D9 23 77 19 10 DC 3E 04 : BA
E9C8 32 5F EB 21 28 00 22 2C : 13
E9D0 E2 16 00 21 43 EA 5E CD : 71
E9D8 81 E7 14 23 3E 0D BA C2 : 66
E9E0 D6 E9 C9 3E 00 32 B8 EC : 9C
E9E8 32 B6 EC 11 F8 EB 2A B4 : A6
E9F0 EC 3E 08 06 00 08 7E FE : BC
E9F8 FF CA 05 EA 4F 3A B6 EC : E3
-----------------------------------
SUM: 89 B1 B7 FC 87 2D 2A 0F 92B2

EA00 3C 32 B6 EC 79 0E 03 ED : 87
```


▶卒論をきれいに書こうとX68000に買いかえました！　何か違うような気が……。
田中　輝未（21）北海道

```
EA08 B0 E5 21 51 EA 4F 09 7E : C7
EA10 13 12 E1 13 13 13 13 13 : 65
EA18 08 3D C2 F5 E9 3A B6 EC : C1
EA20 32 B7 EC 22 B4 EC 21 F9 : B1
EA28 EB 06 08 11 07 00 7E 23 : B2
EA30 32 03 E6 7E 23 32 04 E6 : D8
EA38 CD C4 E4 3A 08 E6 77 19 : 2D
EA40 10 EC C9 00 00 00 00 00 : C5
EA48 05 03 31 00 00 00 64 00 : 9D
EA50 00 05 08 07 20 66 67 20 : 21
EA58 20 72 73 20 74 76 6C 20 : 9B
EA60 20 89 8A 20 20 8B 8C 20 : AA
EA68 20 8D 8E 20 20 77 78 20 : 8A
EA70 20 83 84 20 20 7C 85 86 : EE
EA78 20 66 67 20 68 69 6A 20 : 68
---------------------------------
SUM: D8 4F B0 D7 A1 71 19 AB 4E1B

EA80 20 6B 6C 20 20 77 78 20 : 46
EA88 20 79 7A 7B 20 7C 7D 20 : C7
EA90 6F 70 67 20 20 71 6A 20 : 81
EA98 20 6B 6C 20 20 60 61 20 : 18
EAA0 20 62 63 20 20 64 65 20 : 0E
EAA8 20 77 80 81 20 79 82 20 : D3
EAB0 20 7C 7D 20 20 66 67 20 : 46
EAB8 20 72 73 20 74 75 6E 20 : 9C
EAC0 20 89 8A 20 20 8B 8C 20 : AA
EAC8 20 8D 8E 20 20 77 78 20 : 8A
EAD0 20 83 84 20 20 7E 87 88 : F4
EAD8 20 66 67 20 68 69 6A 20 : 68
EAE0 20 6D 6E 20 20 77 78 20 : 4A
EAE8 20 79 7A 7B 20 7E 7F 20 : CB
EAF0 6F 70 67 20 20 71 6A 20 : 81
EAF8 20 6D 6E 20 20 60 61 20 : 1C
---------------------------------
SUM: 9E 48 4C 17 9C 2B 33 68 8095

EB00 20 62 63 20 20 64 65 20 : 0E
EB08 20 77 80 81 20 79 82 20 : D3
EB10 20 7E 7F 20 8F 90 91 92 : 7F
EB18 93 94 95 96 97 98 99 9A : B4
EB20 9B 9C 9D 9E 9F A0 A1 A2 : F4
EB28 A3 A4 A5 A6 9F A0 A1 A2 : 14
EB30 A3 A4 A5 A6 AB AC AD AE : 44
EB38 AF B0 B1 B2 B3 B4 B5 B6 : 94
EB40 20 20 20 20 00 00 A7 00 : 27
EB48 A8 00 00 00 A9 00 00 00 : 51
EB50 00 00 00 00 AA 20 76 92 : D2
EB58 5E 00 00 00 FF 00 00 04 : 61
EB60 00 00 00 1D 1B 05 06 01 : 44
EB68 1C 19 00 10 00 00 00 F8 : 3D
EB70 03 06 0D 00 10 00 00 00 : 26
EB78 F4 05 16 0D 00 10 00 00 : 2C
---------------------------------
SUM: BC C3 D2 4D 7F DA D8 A3 4CAD

EB80 08 F8 02 17 0E 00 04 00 : 2B
EB88 00 08 FC 02 1A 15 00 10 : 45
EB90 00 00 00 F9 0F 17 0E 00 : 2D
EB98 04 00 00 04 F9 04 1B 16 : 36
EBA0 00 04 00 00 08 FC 08 1B : 2B
EBA8 16 00 04 00 00 04 F1 0D : 1C
EBB0 1A 14 00 04 00 00 00 F6 : 28
EBB8 08 13 06 00 04 00 00 04 : 29
EBC0 FF 03 1A 14 00 04 00 00 : 34
EBC8 0C F3 02 12 04 00 04 00 : 1B
EBD0 00 04 F7 0B 12 04 00 04 : 20
EBD8 00 00 04 FD 03 13 07 00 : 1E
EBE0 08 00 00 04 F7 0B 1D 1B : 46
EBE8 00 08 00 00 00 FA 00 00 : 02
EBF0 00 00 10 00 00 00 00 00 : 10
EBF8 FF 01 00 0D 00 00 00 00 : 0D
---------------------------------
SUM: 56 2E 2F 59 4C 50 4E 67 C7C6

EC00 00 FF FE 00 0D 00 00 00 : 0A
EC08 00 00 FF 00 01 0D 00 00 : 0D
EC10 00 00 00 FF 00 FF 0D 00 : 0B
EC18 00 00 00 00 FF 00 00 07 : 06
EC20 00 00 00 00 00 FF 00 00 : FF
EC28 07 00 00 00 00 00 FF 00 : 06
EC30 00 07 00 00 00 00 00 FF : 06
EC38 00 00 07 00 00 00 00 00 : 07
EC40 FF 00 00 00 04 00 00 00 : 03
EC48 00 FF 00 00 00 04 00 00 : 03
EC50 00 00 FF 00 00 00 04 00 : 03
EC58 00 00 00 FF 00 00 00 04 : 03
EC60 00 00 00 00 FF 0C 04 10 : 1F
EC68 00 00 00 00 FF 0C 04 10 : 1F
EC70 00 00 00 00 FF 0C 04 10 : 1F
EC78 00 00 00 00 FF 0C 04 10 : 1F
---------------------------------
SUM: 06 05 03 FE 0D 3F 20 4A FBAA

EC80 00 00 00 00 FF 18 08 10 : 2F
EC88 00 00 00 00 FF 18 08 10 : 2F
EC90 00 00 00 00 FF 0C 04 10 : 1F
EC98 00 00 00 00 FF 00 00 00 : FF
ECA0 00 00 00 00 FF 00 00 00 : FF
ECA8 00 00 00 00 FF 00 00 00 : FF
ECB0 00 00 00 00 18 F0 FC 01 : 05
ECB8 01 01 00 00 00 00 00 00 : 02
ECC0 00 00 F0 00 00 00 00 00 : F0
ECC8 00 FF 00 00 00 04 00 00 : 03
ECD0 00 00 FF 00 00 00 00 00 : FF
ECD8 00 00 FF 00 00 00 00 00 : FF
ECE0 00 00 FF 00 00 00 00 00 : FF
ECE8 00 00 FF 00 00 00 00 00 : FF
ECF0 00 00 FF 00 00 00 00 00 : FF
ECF8 00 00 FF 00 00 00 00 00 : FF
---------------------------------
SUM: 01 00 EA 00 12 30 10 31 44C5

ED00 FF 00 FF 00 FF 00 FF 00 : FC
ED08 FF 00 FF 00 FF 00 FF 00 : FC
ED10 FF 00 FF 00 FF 00 FF 00 : FC
ED18 FF 00 FF 00 FF 00 FF 00 : FC
ED20 FF 00 FF 00 FF 00 FF 00 : FC
ED28 FF 00 FF 00 FF 00 FF 00 : FC
ED30 FF 00 FF 00 FF 00 FF 00 : FC
ED38 FF 00 FF 00 FF 00 FF 00 : FC
ED40 FF 00 FF 00 FF 00 FF 00 : FC
ED48 FF 00 FF 00 FF 00 FF 00 : FC
ED50 FF 00 FF 00 FF 00 FF 00 : FC
ED58 FF 00 FF 00 FF 00 FF 00 : FC
ED60 FF 00 FF 00 FF 00 FF 00 : FC
ED68 FF 00 FF 00 FF 00 FF 00 : FC
ED70 FF 00 FF 00 FF 00 FF 00 : FC
ED78 FF 00 FF 00 FF 00 FF 40 : 3C
---------------------------------
SUM: F0 00 F0 00 F0 00 F0 40 4E26

ED80 00 FF 00 FF 00 FF 00 FF : FC
ED88 00 FF 00 FF 00 FF 00 FF : FC
ED90 00 FF 00 FF 00 FF 00 FF : FC
ED98 00 FF 00 FF 00 FF 00 FF : FC
EDA0 00 FF 00 FF 00 FF 00 FF : FC
EDA8 00 FF 00 FF 00 FF 00 FF : FC
EDB0 00 FF 00 FF 00 FF 00 FF : FC
EDB8 00 FF 00 FF 00 FF 00 FF : FC
EDC0 00 FF 00 FF 00 FF 00 FF : FC
EDC8 00 FF 00 FF 00 FF 00 FF : FC
EDD0 00 FF 00 FF 00 FF 00 FF : FC
EDD8 00 FF 00 FF 00 FF 00 FF : FC
EDE0 00 FF 00 FF 00 FF 00 FF : FC
EDE8 00 FF 00 FF 00 FF 00 FF : FC
EDF0 00 FF 00 FF 00 FF 00 FF : FC
EDF8 00 FF 00 FF 00 FF 00 BF : BC
---------------------------------
SUM: 00 F0 00 F0 00 F0 00 B0 AC7A

EE00 7E 00 FF 00 FF 00 FF 00 : 7B
EE08 FF 00 FF 00 FF 00 FF 00 : FC
EE10 FF 00 FF 00 FF 00 FF 00 : FC
EE18 FF 00 FF 00 FF 00 FF 00 : FC
EE20 FF 00 FF 00 FF 00 FF 00 : FC
EE28 FF 00 FF 00 FF 00 FF 00 : FC
EE30 FF 00 FF 00 FF 04 FF 00 : 00
EE38 FF 00 FF 00 FF 00 FF 00 : FC
EE40 FF 00 FF 00 FF 00 FF 00 : FC
EE48 FF 4D 0B 02 00 4D 0B 08 : B9
EE50 01 13 04 CA AB 90 7E 08 : A3
EE58 01 31 55 31 55 CB AB CB : 4E
EE60 AB 0E 4F E7 4E BB 4E 5D : A3
EE68 4E 13 4E E8 4D 85 4D 42 : F8
EE70 4D 22 4D FB 3A 79 4C CD : 83
EE78 AB 39 29 57 52 0F EF 08 : BC
---------------------------------
SUM: 68 0D 6E 1E 1E 74 01 4F D3AD

EE80 08 E7 4E BB 4E 5D 4E 13 : 04
EE88 4E E8 4D 85 4D 42 4D 22 : 06
EE90 4D 8C 4C AC 51 0F EF 0C : 2C
EE98 50 CF AB 17 EF E7 4E BB : C0
EEA0 4E 4D 0B 05 01 55 03 0D : 11
EEA8 08 11 08 CA 03 6D AB 07 : 0D
EEB0 EF 31 55 66 AB 31 55 70 : 7C
EEB8 AB 0E 4F 31 55 8E 7D 8E : 27
EEC0 7D FF EE E7 4E BB 4E 5D : 05
EEC8 4E 13 4E E8 4D 85 4D 92 : 48
EED0 7E 2C 56 A1 7D 2C 56 97 : 37
EED8 4F 0E 4F 01 7A 0F 00 01 : 37
EEE0 7A 0F 00 01 7A 01 7A 01 : 80
EEE8 03 0F 03 00 EE 1A 05 24 : 46
EEF0 79 02 0F 00 E0 00 19 14 : 97
EEF8 98 AA 7D 89 15 FF FF FF : 5A
---------------------------------
SUM: 09 DD B9 64 CE AB E0 CD 33F8

EF00 F9 00 00 00 00 00 00 03 : FC
EF08 00 00 00 00 00 00 00 09 : 09
EF10 00 00 00 00 00 00 00 01 : 01
EF18 00 00 00 00 00 00 00 01 : 01
EF20 00 00 00 00 00 00 00 00 : 00
EF28 FF 00 FF 00 FF 00 FF 00 : FC
EF30 FF 00 FF 00 FF 00 FF 00 : FC
EF38 FF 00 FF 00 FF 00 FF 00 : FC
EF40 FF 00 FF 00 FF 00 FF 00 : FC
EF48 FF 00 FF 00 FF 00 FF 00 : FC
EF50 FF 00 FF 00 FF 00 FF 00 : FC
EF58 FF 00 FF 00 FF 00 FF 00 : FC
EF60 FF 00 FF 00 FF 00 FF 00 : FC
EF68 FF 00 FF 00 FF 00 FF 00 : FC
EF70 FF 00 FF 00 FF 00 FF 00 : FC
EF78 FF 00 FF 00 FF 00 FF 40 : 3C
---------------------------------
SUM: EE 00 F5 00 F5 00 F5 4E 07FF

EF80 00 FF 00 FF 00 FF 00 FF : FC
EF88 00 FF 00 FF 00 FF 00 FF : FC
EF90 00 FF 00 FF 00 FF 00 FF : FC
EF98 00 FF 00 FF 00 FF 00 FF : FC
EFA0 00 FF 00 FF 00 FF 00 FF : FC
EFA8 00 FF 00 FF 00 FF 00 FF : FC
EFB0 00 FF 00 FF 00 FF 00 FF : FC
EFB8 00 FF 00 FF 00 FF 00 FF : FC
EFC0 00 FF 00 FF 00 FF 00 FF : FC
EFC8 00 FF 00 FF 00 FF 00 FF : FC
EFD0 00 FF 00 FF 00 FF 00 FF : FC
EFD8 00 FF 00 FF 00 FF 00 FF : FC
EFE0 00 FF 00 FF 00 FF 00 FF : FC
EFE8 00 FF 00 FF 00 FF 00 FF : FC
EFF0 00 FF 00 FF 00 FF 00 FF : FC
EFF8 00 FF 00 FF 00 FF 00 BF : BC
---------------------------------
SUM: 00 F0 00 F0 00 F0 00 B0 AC7A

F000 00 40 00 FF 00 00 FF 00 : 3E
F008 00 FF 00 00 FF 00 00 FF : FD
F010 00 00 FF 00 00 FF 00 00 : FE
F018 00 00 40 FF 00 00 FF 00 : 3E
F020 00 FF 00 00 FF 00 00 FF : FD
F028 00 00 FF 00 00 FF 00 00 : FE
F030 00 C0 00 FF 00 00 FF 00 : BE
F038 00 FF 00 00 FF 00 00 FF : FD
F040 00 00 FF 00 00 FF 00 00 : FE
F048 00 00 C0 FF 00 00 FF 00 : BE
F050 00 FF 00 00 FF 00 00 FF : FD
F058 00 00 FF 00 00 FF 00 00 : FE
F060 00 40 35 00 45 20 FF 00 : D9
F068 00 FF 00 00 FF 00 00 FF : FD
F070 00 00 FF 00 00 FF 00 00 : FE
F078 00 B2 D7 00 BF CD FF 00 : 14
---------------------------------
SUM: 00 ED 07 FC FF E8 FA FB F7A0

F080 00 FF 00 00 FF 00 00 FF : FD
F088 00 00 FF 00 00 FF 00 00 : FE
F090 00 00 C0 00 C0 F0 FF 00 : 6F
F098 00 FF 00 00 FF 00 00 FF : FD
F0A0 00 00 FF 00 00 FF 00 00 : FE
F0A8 00 00 50 00 50 20 FF 00 : BF
F0B0 00 FF 00 00 FF 00 00 FF : FD
F0B8 00 00 FF 00 00 FF 00 00 : FE
F0C0 02 40 00 FF 00 00 FF 00 : 40
F0C8 00 FF 00 00 FF 00 00 FF : FD
F0D0 00 00 FF 00 00 FF 00 00 : FE
F0D8 02 00 40 FF 00 00 FF 00 : 40
F0E0 00 FF 00 00 FF 00 00 FF : FD
F0E8 00 00 FF 00 00 FF 00 00 : FE
F0F0 02 C0 00 FF 00 00 FF 00 : C0
F0F8 00 FF 00 00 FF 00 00 FF : FD
---------------------------------
SUM: 06 FA 4B FD 0A 0B FB FA 2774

F100 00 00 FF 00 00 FF 00 00 : FE
F108 02 00 C0 FF 00 00 FF 00 : C0
F110 00 FF 00 00 FF 00 00 FF : FD
F118 00 00 FF 00 00 FF 00 00 : FE
F120 02 40 35 02 45 20 FF 00 : DD
F128 00 FF 00 00 FF 00 00 FF : FD
F130 00 00 FF 00 00 FF 00 00 : FE
F138 02 B2 D7 02 BF CD FF 00 : 18
F140 00 FF 00 00 FF 00 00 FF : FD
F148 00 00 FF 00 00 FF 00 00 : FE
F150 02 00 C0 02 C0 F0 FF 00 : 73
F158 00 FF 00 00 FF 00 00 FF : FD
F160 00 00 FF 00 00 FF 00 00 : FE
F168 02 00 50 02 50 20 FF 00 : C3
F170 00 FF 00 00 FF 00 00 FF : FD
F178 00 00 FF 00 00 FF 00 00 : FE
---------------------------------
SUM: 0A ED D6 07 0F F7 FB FB 1B46

F180 01 40 00 FF 00 00 FF 00 : 3F
F188 00 FF 00 00 FF 00 00 FF : FD
F190 00 00 FF 00 00 FF 00 00 : FE
F198 01 00 C0 FF 00 00 FF 00 : BF
F1A0 00 FF 00 00 FF 00 00 FF : FD
F1A8 00 00 FF 00 00 FF 00 00 : FE
F1B0 01 C0 00 FF 00 00 FF 00 : BF
F1B8 00 FF 00 00 FF 00 00 FF : FD
F1C0 00 00 FF 00 00 FF 00 00 : FE
F1C8 01 00 40 FF 00 00 FF 00 : 3F
F1D0 00 FF 00 00 FF 00 00 FF : FD
F1D8 00 00 FF 00 00 FF 00 00 : FE
F1E0 00 B0 18 00 40 18 FF 00 : 1F
F1E8 00 FF 00 00 FF 00 00 FF : FD
F1F0 00 00 FF 00 00 FF 00 00 : FE
F1F8 00 B0 18 00 40 D8 FF 00 : DF
---------------------------------
SUM: 04 5B 2B FC 7B EB FA FB 9A65
```

▶あのー，チオビタ飲んで，愛情１冊ください！　ないんですか？ それじゃあ，ほかをさがします。ウソウソ。
筒井　圭一朗（16）埼玉県

```
F200 00 FF 00 00 FF 00 00 FF : FD
F208 00 00 FF 00 00 FF 00 00 : FE
F210 00 00 40 00 00 B0 00 FF 00 : EF
F218 00 FF 00 00 FF 00 00 FF : FD
F220 00 00 FF 00 00 FF 00 00 : FE
F228 00 00 B0 00 00 50 FF 00 : FF
F230 00 FF 00 00 FF 00 00 FF : FD
F238 00 00 FF 00 00 FF 00 00 : FE
F240 00 20 40 00 10 48 00 E0 : 98
F248 D8 FF 00 00 FF 00 00 FF : D5
F250 00 00 FF 00 00 FF 00 00 : FE
F258 02 B0 18 02 40 18 FF 00 : 23
F260 00 FF 00 00 FF 00 00 FF : FD
F268 00 00 FF 00 00 FF 00 00 : FE
F270 02 B0 18 02 40 D8 FF 00 : E3
F278 00 FF 00 00 FF 00 00 FF : FD
--------------------------------
SUM: DC BA 1B 04 3A 83 FC DA 750A

F280 00 00 FF 00 00 FF 00 00 : FE
F288 02 40 00 02 B0 00 FF 00 : F3
F290 00 FF 00 00 FF 00 00 FF : FD
F298 00 00 FF 00 00 FF 00 00 : FE
F2A0 02 00 B0 02 00 50 FF 00 : 03
F2A8 00 FF 00 00 FF 00 00 FF : FD
F2B0 00 00 FF 00 00 FF 00 00 : FE
F2B8 02 20 40 02 10 48 02 E0 : 9E
F2C0 D8 FF 00 00 FF 00 00 FF : D5
F2C8 00 00 FF 00 00 FF 00 00 : FE
F2D0 01 40 00 01 C0 00 FF 00 : 01
F2D8 00 FF 00 00 FF 00 00 FF : FD
F2E0 00 00 FF 00 00 FF 00 00 : FE
F2E8 00 30 30 00 30 D0 02 48 : AA
F2F0 00 FF 00 00 FF 00 00 FF : FD
F2F8 00 00 FF 00 00 FF 00 00 : FE
--------------------------------
SUM: DF CB 1A 07 AB 62 01 23 3427

F300 02 B0 00 00 30 28 00 30 : 3A
F308 D8 FF 00 00 FF 00 00 FF : D5
F310 00 00 FF 00 00 FF 00 00 : FE
F318 00 30 30 00 30 D0 01 48 : A9
F320 00 FF 00 00 FF 00 00 FF : FD
F328 00 00 FF 00 00 FF 00 00 : FE
F330 01 B0 00 00 30 28 00 30 : 39
F338 D8 FF 00 00 FF 00 00 FF : D5
F340 00 00 FF 00 00 FF 00 00 : FE
F348 02 B0 00 02 30 28 02 30 : 3E
F350 D8 FF 00 00 FF 00 00 FF : D5
F358 00 00 FF 00 00 FF 00 00 : FE
F360 02 20 20 02 20 E0 02 28 : 6E
F368 00 FF 00 00 FF 00 00 FF : FD
F370 00 00 FF 00 00 FF 00 00 : FE
F378 02 40 40 02 40 C0 02 C0 : 46
--------------------------------
SUM: 91 9B 8B 06 1B E3 07 BB 2AC5

F380 40 02 C0 C0 FF 00 00 FF : C0
F388 00 00 FF 00 00 FF 00 00 : FE
F390 02 00 40 02 40 00 02 C0 : 46
F398 00 02 00 C0 FF 00 00 FF : C0
F3A0 00 00 FF 00 00 FF 00 00 : FE
F3A8 02 40 00 02 20 20 02 20 : A6
F3B0 E0 02 D8 18 02 D8 E8 FF : 93
F3B8 00 00 FF 00 00 FF 00 00 : FE
F3C0 00 40 00 00 00 40 00 C0 : 40
F3C8 00 00 00 C0 FF 00 00 FF : BE
F3D0 00 00 FF 00 00 FF 00 00 : FE
F3D8 00 30 00 00 20 20 00 20 : 90
F3E0 E0 00 D8 18 00 D8 E8 FF : 8F
F3E8 00 00 FF 00 00 FF 00 00 : FE
F3F0 00 F0 30 00 F0 D0 00 DC : BC
F3F8 12 00 DC EE FF 00 00 FF : DA
--------------------------------
SUM: 16 A6 B7 62 6E FB D4 96 DDD5

F400 00 00 FF 00 00 FF 00 00 : FE
F408 00 10 30 00 10 D0 00 24 : 44
F410 12 00 24 EE FF 00 00 FF : 22
F418 00 00 FF 00 00 FF 00 00 : FE
F420 01 24 12 01 24 EE 01 10 : 5B
F428 30 01 10 D0 FF 00 00 FF : 0F
F430 00 00 FF 00 00 FF 00 00 : FE
F438 01 C0 00 01 D0 00 00 D8 : 6A
F440 00 00 E8 00 FF 00 00 FF : E6
F448 00 00 FF 00 00 FF 00 00 : FE
F450 02 40 00 02 32 00 00 20 : 96
F458 00 00 18 00 FF 00 00 FF : 16
F460 00 00 FF 00 00 FF 00 00 : FE
F468 00 18 18 00 00 18 00 E8 : 30
F470 18 00 18 E8 00 00 E8 00 : 00
F478 E8 E8 FF 00 00 FF 00 00 : CE
--------------------------------
SUM: 46 35 A0 AA 32 D0 E9 10 FD54

F480 00 18 18 00 18 00 00 18 : 60
F488 E8 00 E8 18 00 E8 00 00 : D0
F490 E8 E8 FF 00 00 FF 00 00 : CE
F498 02 20 20 02 20 E0 02 F0 : 36
F4A0 2C 02 F0 D4 02 D0 15 02 : DB
F4A8 D0 EB FF 00 00 FF 00 00 : B9
F4B0 01 25 1B 01 1B 25 01 DB : 5E
F4B8 E5 01 E5 DB FF 00 00 FF : A4
F4C0 00 00 FF 00 00 FF 00 00 : FE
F4C8 01 40 35 01 45 20 FF 00 : DB
F4D0 00 FF 00 00 FF 00 00 FF : FD
F4D8 00 00 FF 00 00 FF 00 00 : FE
F4E0 01 B2 D7 01 BF CD FF 00 : 16
F4E8 00 FF 00 00 FF 00 00 FF : FD
F4F0 00 00 FF 00 00 FF 00 00 : FE
F4F8 01 00 C0 01 C0 F0 FF 00 : 71
--------------------------------
SUM: B7 23 D7 CD 16 95 15 E2 C7CF

F500 00 FF 00 00 FF 00 00 FF : FD
F508 00 00 FF 00 00 FF 00 00 : FE
F510 01 00 50 01 50 20 FF 00 : C1
F518 00 FF 00 00 FF 00 00 FF : FD
F520 00 00 FF 00 00 FF 00 00 : FE
F528 01 B0 18 01 40 18 FF 00 : 21
F530 00 FF 00 00 FF 00 00 FF : FD
F538 00 00 FF 00 00 FF 00 00 : FE
F540 01 B0 18 01 40 D8 FF 00 : E1
F548 00 FF 00 00 FF 00 00 FF : FD
F550 00 00 FF 00 00 FF 00 00 : FE
F558 01 40 00 01 B0 00 FF 00 : F1
F560 00 FF 00 00 FF 00 00 FF : FD
F568 00 00 FF 00 00 FF 00 00 : FE
F570 01 00 B0 01 00 50 FF 00 : 01
F578 00 FF 00 00 FF 00 00 FF : FD
--------------------------------
SUM: 05 9A 2B 05 7A 5B FB FA A725

F580 00 00 FF 00 00 FF 00 00 : FE
F588 01 20 40 01 10 48 01 E0 : 9B
F590 D8 FF 00 00 FF 00 00 FF : D5
F598 00 00 FF 00 00 FF 00 00 : FE
F5A0 01 40 00 01 00 40 01 C0 : 43
F5A8 00 01 00 C0 FF 00 00 FF : BF
F5B0 00 00 FF 00 00 FF 00 00 : FE
F5B8 01 30 00 01 20 20 01 20 : 93
F5C0 E0 01 D8 18 01 D8 E8 FF : 91
F5C8 00 00 FF 00 00 FF 00 00 : FE
F5D0 01 18 18 01 00 18 01 E8 : 33
F5D8 18 01 18 E8 01 00 E8 01 : 03
F5E0 E8 E8 FF 00 00 FF 00 00 : CE
F5E8 01 18 18 01 18 00 01 18 : 63
F5F0 E8 01 E8 18 01 E8 00 01 : D3
F5F8 E8 E8 FF 00 00 FF 00 00 : CE
--------------------------------
SUM: 8D 93 42 DD 49 7A D5 BF D121

F600 00 20 20 00 20 E0 00 F0 : 30
F608 2C 00 F0 D4 00 D0 15 00 : D5
F610 D0 EB FF 00 00 FF 00 00 : B9
F618 01 20 20 01 20 E0 01 F0 : 33
F620 2C 01 F0 D4 01 D0 15 01 : D8
F628 D0 EB FF 00 00 FF 00 00 : B9
F630 02 30 15 02 30 EB 02 00 : 66
F638 2C 02 00 D4 02 D0 15 02 : EB
F640 D0 EB FF 00 00 FF 00 00 : B9
F648 00 30 15 00 30 EB 00 00 : 60
F650 2C 00 00 D4 00 D0 15 00 : E5
F658 D0 EB FF 00 00 FF 00 00 : B9
F660 01 30 15 01 30 EB 01 00 : 63
F668 2C 01 00 D4 01 D0 15 01 : E8
F670 D0 EB FF 00 00 FF 00 00 : B9
F678 02 28 15 01 28 EB 02 00 : 55
--------------------------------
SUM: F2 93 6F 29 FC 77 6F E4 1CFC

F680 2C 01 00 D4 01 D8 15 01 : F0
F688 D8 EB FF 00 00 FF 00 00 : C1
F690 02 28 15 02 28 EB 02 00 : 56
F698 2C 02 00 D4 02 D8 15 02 : F3
F6A0 D8 EB 00 0D 12 FF 00 00 : E1
F6A8 02 28 15 02 28 EB 02 00 : 56
F6B0 2C 02 00 D4 02 D8 15 02 : F3
F6B8 D8 EB 00 EF ED FF 00 00 : 9E
F6C0 01 28 15 01 28 EB 01 00 : 53
F6C8 2C 02 00 D4 01 D8 15 02 : F2
F6D0 D8 EB 00 EF ED FF 00 00 : 9E
F6D8 02 28 15 01 28 EB 02 00 : 55
F6E0 2C 01 00 D4 01 D8 15 01 : F0
F6E8 D8 EB 00 0D 13 FF 00 00 : E2
F6F0 00 10 05 00 20 0D 00 30 : 72
F6F8 1D 00 F0 05 00 E0 0D 00 : FF
--------------------------------
SUM: 38 4F 48 27 C6 CC 7D 38 A7DD

F700 D0 1D FF 00 00 FF 00 00 : EB
F708 00 10 FB 00 20 F3 00 30 : 4E
F710 E3 00 F0 FB 00 E0 F3 00 : A1
F718 D0 E3 FF 00 00 FF 00 00 : B1
F720 02 10 FB 02 20 F3 02 30 : 54
F728 E3 02 F0 FB 02 E0 F3 02 : A7
F730 D0 E3 FF 00 00 FF 00 00 : B1
F738 02 10 05 02 20 0D 02 30 : 78
F740 1D 02 F0 05 02 E0 0D 02 : 05
F748 D0 1D FF 00 00 FF 00 00 : EB
F750 00 10 0B 00 20 F3 00 30 : 5E
F758 E3 00 F0 F5 00 E0 0D 00 : B5
F760 D0 1D FF 00 00 FF 00 00 : EB
F768 02 10 0B 02 20 F3 02 30 : 64
F770 E3 02 F0 F5 02 E0 0D 02 : BB
F778 D0 1D FF 00 00 FF 00 00 : EB
--------------------------------
SUM: 8F 90 BB EB A6 33 13 F6 78AB

F780 02 40 00 02 00 40 02 C0 : 46
F788 00 02 00 C0 00 E8 E8 00 : 92
F790 E8 18 00 18 18 00 18 E8 : 30
F798 02 40 00 02 00 40 02 C0 : 46
F7A0 00 02 00 C0 02 E8 E8 02 : 96
F7A8 E8 18 02 18 18 02 18 E8 : 34
F7B0 01 40 00 01 00 40 01 C0 : 43
F7B8 00 01 00 C0 02 E8 E8 02 : 95
F7C0 E8 18 02 18 18 02 18 E8 : 34
F7C8 01 FB 10 01 F3 20 01 E3 : 04
F7D0 30 01 FB F0 01 F3 E0 01 : F1
F7D8 E3 D0 FF 00 00 FF 00 00 : B1
F7E0 01 05 10 01 0D 20 01 1D : 62
F7E8 30 01 05 F0 01 0D E0 01 : 15
F7F0 1D D0 FF 00 00 FF 00 00 : EB
F7F8 00 E8 27 00 30 E9 00 20 : 48
--------------------------------
SUM: 1F 97 49 6F 7E A3 C7 1E 2E82

F800 34 00 D0 E7 FF 00 00 FF : E9
F808 00 00 FF 00 00 FF 00 00 : FE
F810 02 E8 27 00 30 E9 02 20 : 4C
F818 34 00 D0 E7 02 FD E0 FF : C9
F820 00 00 FF 00 00 FF 00 00 : FE
F828 02 E8 27 02 30 E9 02 20 : 4E
F830 34 02 D0 E7 02 FD E0 FF : CB
F838 00 00 FF 00 00 FF 00 00 : FE
F840 01 E8 27 02 30 E9 02 20 : 4D
F848 34 01 D0 E7 01 FD E0 FF : C9
F850 00 00 FF 00 00 FF 00 00 : FE
F858 01 E8 27 01 30 E9 01 20 : 4B
F860 34 01 D0 E7 01 FD E0 02 : CC
F868 2D 15 FF 00 00 FF 00 00 : 40
F870 02 15 25 00 2D 00 00 00 : 69
F878 CF 00 E4 30 02 1E D2 00 : D5
--------------------------------
SUM: 08 CE B0 B8 F4 B1 59 7E 1B4D

F880 C8 E2 00 D8 12 FF 00 00 : 93
F888 02 15 25 02 2D 00 02 00 : 6D
F890 CF 02 E4 30 02 1E D2 02 : D9
F898 C8 E2 02 D8 12 FF 00 00 : 95
F8A0 01 15 25 02 2D 00 01 00 : 6B
F8A8 CF 01 E4 30 01 1E D2 02 : D7
F8B0 C8 E2 02 D8 12 FF 00 00 : 95
F8B8 01 15 25 01 2D 00 01 00 : 6A
F8C0 CF 01 E4 30 01 1E D2 01 : D6
F8C8 C8 E2 01 D8 12 FF 00 00 : 94
F8D0 02 18 00 02 00 18 02 12 : 48
F8D8 12 02 15 F0 02 F0 15 02 : 22
F8E0 E5 E5 02 DE ED FF 00 00 : 96
F8E8 02 18 00 02 00 18 02 12 : 48
F8F0 12 02 15 F0 02 F0 15 01 : 21
F8F8 E5 E5 01 DE ED FF 00 00 : 95
--------------------------------
SUM: 83 C9 4D 95 B1 64 A8 2C D660

F900 02 10 FB 02 20 F3 02 30 : 54
F908 E3 02 F0 FB 02 E0 F3 02 : A7
F910 D0 E3 02 F8 20 02 08 20 : F7
F918 01 10 FB 01 20 F3 01 30 : 51
F920 E3 01 F0 FB 01 E0 F3 01 : A4
F928 D0 E3 02 F8 20 02 08 20 : F7
F930 02 10 0B 02 20 F3 02 30 : 64
F938 E3 02 F0 F5 02 E0 0D 02 : BB
F940 D0 1D 02 F5 22 02 0A DE : F0
F948 02 10 0B 02 20 F3 01 30 : 63
F950 E3 02 F0 F5 02 E0 0D 01 : BA
F958 D0 1D 01 F5 22 01 0A DE : EE
F960 01 10 0B 01 20 F3 01 30 : 61
F968 E3 01 F0 F5 01 E0 0D 01 : B8
F970 D0 1D 01 F5 22 01 0A DE : EE
F978 C1 C1 D1 E1 C9 FF FF FE : F9
--------------------------------
SUM: 48 36 A0 8D 17 26 41 CF 2744

F980 00 00 00 00 00 00 00 00 : 00
F988 00 00 00 01 02 02 02 02 : 09
F990 02 01 21 1B FB 06 0F 11 : 60
F998 06 00 CD FA F7 E6 1F 77 : 40
F9A0 23 CD FA F7 E6 1F 77 3E : 9B
F9A8 00 23 77 CD FA F7 E6 03 : 41
F9B0 3C D9 47 3E 01 07 10 FD : AF
F9B8 D9 23 77 19 10 DC 3E 04 : BA
F9C0 32 11 FB 21 28 00 22 C8 : 71
F9C8 F1 16 00 21 F7 F9 5E CD : 43
F9D0 79 F7 14 23 3E 0D BA C2 : 6E
F9D8 CE F9 21 AC FB 06 08 11 : AE
F9E0 07 00 7E 23 32 13 F6 7E : 61
F9E8 23 32 14 F6 CD D4 F4 3A : 2E
F9F0 18 F6 77 19 10 EC C9 00 : 63
F9F8 00 00 00 00 05 03 31 00 : 39
--------------------------------
SUM: EC 2C 56 74 51 C9 01 EC DE94
```

X1/X1turbo用 ©TOKUMA MUSIC PUBLISHER Co.LTD.
天空の城ラピュタより パズーとシータ

Nagase Hideaki
永瀬 秀昭

X68000用 ©SEGA
ギャラクシーフォースより Beyond the Galaxy

Nishikawa Zenji
西川 善司

めっきり寒くなってきましたね。今年もあますところあと少し。有意義に過ごしたいものですね。さて，今年のOh!X LIVEの締めくくりは最後を飾るにふさわしい映画&ゲームミュージックです。来年も皆さんの投稿をお待ちしております。

人気の宮崎作品より

X1用にはアニメ映画「天空の城ラピュタ」からのイメージ・アルバム「空から降ってきた少女」より，「パズーとシータ」です。宮崎駿監督の作品としては最新作「魔女の宅急便」も大ヒットしましたが，「風の谷のナウシカ」と並びこのラピュタにも根強いファンがいるようです。Oh!X LIVEに寄せられる投稿の中でも魔女の宅急便がまだ送られてきてないのに対し，ラピュタは何作か拝聴させていただきました(催促[2])。全体的に見るとアニメソングは少ないですね。自他共に認めるオタクの○○さんや，××さん，自分では否定しているけど立派なオタクの△△さん等をかかえるOh!Xのスタッフ事情からするとあまりに少ないアニメ関係。必修課題，最重要ポイントはアニメだ！(うーん，受験シーズン到来)。ついでに本誌に載るためのポイントをいくつかチェックしてみましょう。題して「こうすれば君もOh!X LIVEの覇者になれる！」

その1：いい曲を作ろう！(あったりまえだ)。聞いてみて，思わずもう一度聞きたくなるような曲がベスト。先月号のOb-la-di, Ob-la-daなんかは代表例。何度聞いても楽しめます(おかげで私はビートルズのCDを買ってしまった)。

その2：選曲はきっちりと！(意外とムズいかも)。確かに好きな曲を作るのが一番よいのですが，自分のレベルにあった中の好きな曲にしたほうが出来はよいのです。本当に一番好きな曲は自分のレベル試しに何度も作ってみよう。

その3：投稿するときにはメディア(ディスクやカセット)にきちんと作品名とあなたの名前を書いておこう！(意外と書いていないものも見受けられる)。たとえば先月号のメタルホークなんぞは時期を同じくしてX68000・OPMA用に4つの作品が届きました。そうしたら聞き比べてみたくなるのが人情というものです。もちろん，こちらできっちり管理をしていますので間違えることはありませんが，念には念をというわけです。最近では1日おきでX1用の「RUNNER」が届きました。迷子のディスクに注意しましょう。

その4：ゲームミュージックは倍率がとっても高い！　上の話と少し重なりますがゲームミュージックは山のように届きます。怖いことに，いまでも週に数本の割合でYs(1〜3)の投稿があります。もし，これからYsを投稿して採用されたのなら，本当に素晴らしいアレンジをした人でしょう。

その5：聞いてほしかったらディスクにシステムを入れてこい！(これが一番言いたかったりする)。はっきり言って膨大な投稿作品たちと格闘をする担当者(よーするに私)にとって，いちいちディスクをとっかえひっかえする苦労は計りしれません(自分で言うなよ)。X1用とだけ書いてきて，祝版でもないしMIDI＋でもない，Music BASICでもなし，Z-BASICでもなく，もちろんS-OS用のFM音源システムでもなく，首をひねりながら最後に試した普通のBASIC用のPSGで作ってあった作品には溜息が出ました(いまでは笑い話になっているけどね)。とりあえず，あなたがその作品を作ったときのシステムディスクをまるごとコピーして，あとから作品だけをセーブしても結構ですから(オート・スタートなんて贅沢はいいません)，その作品が聞けるシステムだけは入れてください。担当者としての切実なお願いです。なお，逆にテープの場合はプログラムだけのほうがありがたかったりします。ちなみに今月号のラピュタには，きちんとMusicBASICが組み込まれていました。

天空の城ラピュタ

話をラピュタに戻しますが，この作品が届いたのは89年の2月ですので，ほぼ⅔年ぶりの採用となったわけです。素晴らしい作品は1年後でもちゃんと載ります。

曲にあっているとはいえ，ちょっとビブラートがかかりすぎてるかな？　とも思えましたが良しとしましょう。すべてはバランスですぞ。　(S.K.)

ギャラクシーフォース

こんにちは，善司です。11月号の進藤氏の「メタルホーク」はもう皆さん入力しましたか？　あれは，すごい出来です。ぜひ入力して聞いてみましょう。(私はあれを聴いたとき，初め言葉を失った。『うまい，うますぎる……』 ©十万石まんじゅう)彼に刺激されて私はOPMAドライバ専用にギャラクシーフォースの「Beyond the Galaxy」をプログラムしてみました。かなり力を入れて作ったのでぜひとも聞いてみてください。

今回のプログラムの目玉(にしてはたいしたことはないけど)は簡易ベンドサブルーチンです。これは，KF(キーフラクショ

ン）を滑らかに変化させるMMLを生成するルーチンです（いやぁ，立川君や進藤君のYコマンド攻撃はすごかった……）。

関数名は「S( )」で～，

Z=S("A",12,252,0,1) (Zは文字型変数）のようにして使います。この例だとA＃(252) からA(0)まで12個分の精度で MML をチャンネル1用に作成します。この例ではピッチダウンですが，0と252を入れ換えればアップも可能です。注意点としては誤差のため，終了時のKFが指定通りにならないことがあります。ですから，一度ダイレクトで関数を実行し，どんなMMLが生成されたか，確認したほうがよいでしょう。

プログラムはハイハットやライドシンバル，カウベルがパート数不足のため，あちこちのチャンネルに割り込んでいます。ハイハットは一部原曲とは違ったパターンのところがあります。これはライドシンバルのボリュームが小さいので，これをバックアップするためにそうしました。それとドラムスに凝ってみたんですが少々文字列群がでかくなりすぎましたかね。はははは。

最後にX68000の設計者に一言「AD PCMになんでボリューム付けなかったんだよーっ!!」(バスドラが聞こえない)。(Z.N.)

ギャラクシーフォース

MusicBASICは1988年12月号で発表されたX1用MML，OPMAは1989年4月号で発表されたFM音源とPCM音源を同期させるX68000用ミュージックドライバです（OPMAがなくてもFM音源の演奏は可能です）。

日本音楽著作権協会(出)許諾第8971746-901号

リスト1　パズーとシータ

```
10 '
20 '    THE CASTLE IN THE SKY 「LAPUTA」
30 '        ﾖﾘ  「ﾊﾟｽﾞｰ ﾄ ｼｰﾀ」
40 '
50 '    MUSIC FROM  JOE HISAISHI
60 '
70 '    PROGRAM BY H.NAGASE    '88.12.26
80 '
90 TEMPO0:PLAY "X":M=0:GOTO 220
100 LABEL "A"
110 PLAY M$+":"+M$+":"+C1$+":"+C1$+":";
120 IF M=1 THEN 160
130 PLAY C2$+":"+C2$+":"+B$+":"+B$+":";
140 PLAY C2$+":"+C2$+":"+B$
150 RETURN
160 PLAY C2$+":"+C2$+":"+B$+":"+B$
170 RETURN
180 LABEL "B"
190 PLAY M$+":"+M$+":"+C1$+":"+C1$+":";
200 PLAY C2$+":"+C2$+":"+C3$+":"+B$
210 RETURN
220 PLAY "T116I4P1K10V13O4L8CD:I4P2K0V13O4L8CD:I4P1K0V10O3L8RR:I
4P2K10V10O3L8RR:";
230 PLAY "I4P1K15V10O3L8RR:I4P2K0V10O3L8RR:I3P1K10V10O4L8RR:I3P2
K0V10O4L8RR:";
240 PLAY "O4L8RRV11K0Y7,56:O4L8RRV11K3:O3L8RRV11"
250 M$="S4,10,11,0":C1$=M$:C2$=M$:B$=M$:"A"
260 ' GOTO910
270 '
280 M1$="E-4DE-4G4D8&D2.<G4>C4<B->C4E-4<B-&B-2.G4
290 M2$="A-4.GA->E-4.<G4.FG>E-4.D4.<AA4>D4
300 M$=M1$:C1$="G1&G1E-2&E-GFE-E-1
310 C2$="G1&G1E-2&E-E-DC<B-1>
320 B$="C1<B-1A-1G1
330 "A"
340 M$=M2$:C1$="R2E-4DCC2.C4F+4.RF+4GA
350 C2$="R2E-4DCC2.C4F+4RD4EF+
360 B$="F2>F4D4E-1D1
370 "A"
380 M$="D2R4CD"+M1$
390 C1$="<G2B2>G1&G1E-2&E-GFE-E-DE-DGFE-4
400 C2$="G2GFE-DG1&G1E-2&E-E-DCC<B->C<B->E-<B-G4>
410 B$="G2GFE-DC1<B-1A-1G1
420 "A"
430 M$="A-4>E-D&DE-4.F4GE-&E-2E-DC4D4<B4>C2R4E-F
440 C1$="F4.A-&A-B->C4<F4G2.A-2G2G2R2
450 C2$="F4.A-&A-B->C4<F4G2.C2DE-F4E-2R2
460 B$=">F2.E-DC2.DE-F2G2C2R2
470 "A"
480 M$="G4FG4B-4F&F2R4<B-4>E-4DE-4G4G&G2R4G4>C2.<B-A-B-2E-4.E-
490 C1$="I4O3RB-RB-RB-RB-RB-RB-RB-RB-RGRGRGRGRGRGRGRGRE-RE-RE-RE
-RE-RE-RE-RE-
500 C2$="I4O3RGRGRGRGRFRFRFRFRE-RE-RE-RE-RDRDRDRDRCRCRCRCR<B-RB-
RB-RB->
510 B$="L4E-E-E-E-DDDDCCCC<B-B-B-B-A-A-A-A-GGGG
520 "A"
530 M$="L4A-GFE-L8DE-FD<B-4>B-A-G1&G2.CD
540 C1$="I4L4>C<B-A-GL8F2.R4<BAG4>FE-D4A-GF4BAG4
550 C2$="I4L4C<B-A-GL8F2.R4BAG4>DC<B4>FE-D4GFD4
560 B$="FGA-AB-1B1G1
570 "A"
580 M$="E-4DE-4G4D&D2.<G4>C4<B->C4E-4<B-&B-2R4G4A-4>E-D4E-4.
590 C1$="G1D1G2&GGFFE-1A-2&A-B->C4<
600 C2$="E-1<B-1>E-2&E-E-DC<B-1>C1
610 B$="C2&>C2<B-1A-1G1F2.E-8D8
620 "A"
630 M$="F4GE-&E-2E-DC4D4<B4>=2C1=0
640 C1$="G1A-2G2=2G1=0
650 C2$="G1F2DE-F4=2E1=0
660 B$="C2.D8E-8F2G2=2C1=0
670 "A"
680 '
690 M$="L8R2R4I6V14O5CDE-4.DE-G4.D2R4<G4>C4.<B->CE-4.<B-2R4G4
700 C1$="L1RI6V11O4GGE-E-
710 C2$="L1RI6V11O4E-DC<B-
720 B$="L1RI6V11O3C<B-A-G
730 M=1:"A"
740 M$="A-4.GA->E-4.<G4.FG>E-4.D4.<AA4>D4D2R4CD
750 C1$="E-CL32RRRC&C8&C2.L1D
760 C2$=">CGO3L16RA&A8&A2.L1G
770 C3$="FE-O3L32RF+&F+&F+&F+8&F+2.L1G
780 B$="FE->O3D1L1<G
790 "B"
800 M$="E-4.DE-G4.D2R4<G4>C4.<B->CE-4.<B-2R4G4A-4>E-D&D4E-4
810 C1$="GGL16RE-&E-8&E-2.L32RRRF&F8&F2.L16RE-&E-8&E-2.
820 C2$=">E-DL32RC&C&C&C8&C2.L16RE-&E-8&E-2.L32RC&C&C&C8&C2.
830 C3$=">C<B->A-1L32RB-&B-&B-&B-8&B-2.L1F
840 B$=">C<B-A-GF
850 "B"
860 M$="F4GE-&E-2E-DC4D4<B4>C2R4E-F
870 C1$="O3C4G4>E-2<A-2F2>L16RC&C8&C2R4O4
880 C2$="O4C4G4>E-2<A-2F2<L32RG&G&G&G8&G2R4O3
890 B$="L1CF2G2C2.R4O3
900 "A"
910 M$="G4.FGB-4F8&F2.R8<B->E-4.DE-G4.G2R4G4
920 C1$="L4CB->G2<DB->F2<CG>E-2<<B->G>D2
930 C2$=C1$
940 B$="CDC<B-
950 "A"
960 M$="O6L16RC&C8&C2<L8B-A-B-4.E-E-2L4A-GFE-
970 C1$="O5L32RG&G&G&G8&G2R4F4.<B-8B-2>C1
980 C2$="O5E-2.R4O3G1F1
990 B$="O2A-4>E-4>C2<<L1GF
1000 "A"
1010 M$="L8G2&G4<B4>CDE-4.DE-G4D&D2.R<G>
1020 C1$="L8O4RG&G2.&G4O3C1<B-1
1030 C2$="L8O4R16D.&D2.&D4L4<CG>E-2<<B->G>D2
1040 C3$="L8O4R32C16.&C&C2.&C4L4<CG>E-2<<B->G>D2
1050 B$="L8O3G1&G4C1<L1B-
1060 "B"
1070 M$="C4.<B->CE-4.<B-2.G4
1080 A$="L4A->E->C2<<G>E-B-2
1090 C1$="O2"+A$
1100 C2$="O3"+A$
1110 B$="O2A-G
1120 "A"
1130 M$="A-4>E-D&D4E-4F4GE-&E-2E-DC4D4<B4
1140 C1$="O2F>CA-2CG>E-2A-2F2
1150 C2$=C1$
1160 B$="F>CF2G2
1170 "A"
1180 A1$="O5L32RRRRC&
1190 A2$="O4L32RRRG&
1200 A3$="P3K5O4L32RRRE&
1210 A4$="P3K5O4L32RRC&
1220 A5$="P3K5O3L32RG&
1230 A6$="P3K5O2C&
1240 PLAY A1$+":"+A1$+":"+A2$+":"+A2$+":"+A3$+":"+A4$+":"+A5$+":
"+A6$
1250 PLAY "X"
```

リスト2 Beyond the Galaxy

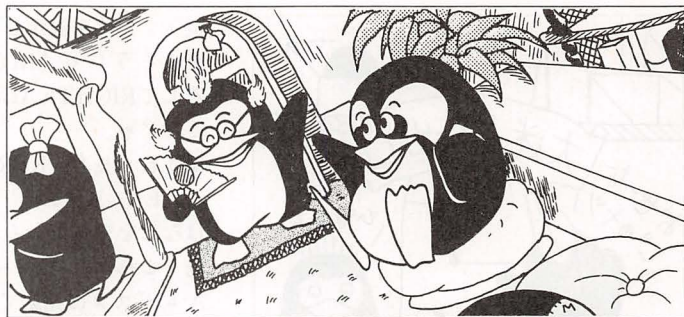
```
10 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
20 /*            G A L A X Y   F O R C E          /*
30 /*                                             /*
40 /*      B E Y O N D   T H E   G A L A X Y      /*
50 /*                                             /*
60 /*              (C) S.S.T. SEGA                /*
70 /*                                             /*
80 /*          Arranged by Z.Nishikawa            /*
90 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
100 m_init()
110 dim char  v(4,10)={        /*       E . B A S S
120 /*     AF  OM  WF  SY SP  PMD AMD PMS AMS PAN /
130        58, 15,  0,  0,  0,  0,  0,  0,  0,3,  0,
140 /*     AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AM-EN
150        31, 10,  7,  8,  2, 33,  0,  0,  7,  0,0,
160        21,  8,  8,  7,  5, 23,  3,  7,  7,  2,0,
```

```
170         31,  5,  6,  7,  1, 37,  0,  0,  3,  0,0,
180         31,  8,  6,  7,  5,  0,  0,  1,  7,  0,0}
190 m_vset(1,v)
200   v={ /*  H I H A T   O P (コノ オトハ Oh! X 10ガツゴウ ケイサイ ノ
210 /*   AF  OM  WF  SY  SP PMD AMD PMS AMS PAN   /
     'Top Landing' by Hideki Nishimoto
220        59, 15,  3,  0,200, 50,  0,  6,  0,  1,  0,  /*
                                         ノ モノ ヲ edit シタ モノ デス。)
230 /*   AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2   AM-EN
240        31,  5,  4,  2,  2, 21,  0,  3,  3,  3,  0,
250        31,  3,  2,  1,  2, 22,  0,  1,  0,  2,  0,
260        31, 13, 11,  2,  2, 28,  0, 14,  0,  3,  0,
270        31, 19,  5,  4,  3,  0,  0, 14,  7,  3,  0}
280 m_vset(2,v)
290   v={          /*       H I H A T   C L O S E
300 /*      AF  OM  WF  SY SP  PMD AMD PMS AMS PAN /
310         61, 15,  0,  0,  0,  0,  0,  0,  0,1,  0,
320 /*      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AM-EN
330         26, 10,  5, 10, 10,  0,  2, 14,  0,  3,0,
340         31, 20, 11, 15, 11,127,  1,  3,  7,  1,0,
350         28, 24, 10, 15, 10,  2,  0,  4,  7,  1,0,
360         19, 21, 12, 15, 11,  0,  0,  2,  7,  1,0}
370 m_vset(3,v)
380   v={          /*       S T R I N G S
390 /*      AF  OM  WF  SY SP  PMD AMD PMS AMS PAN /
400         58, 15,  0,  0,  0,  0,  0,  0,  0,3,  0,
410 /*      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AM-EN
420         19,  5,  0,  5,  1, 30,  0,  2,  0,  0,1,
430         24,  4,  0,  5,  1, 39,  0,  2,  0,  0,1,
440         28,  3,  0,  5,  1, 32,  0,  2,  0,  0,1,
450         20,  4,  0,  6,  2,  0,  0,  2,  7,  0,1}
460 m_vset(4,v)
470   v={          /*       E. P I A N O
480 /*      AF  OM  WF  SY SP  PMD AMD PMS AMS PAN /
490         52, 15,  0,  0,  0,  0,  0,  0,  0,3,  0,
500 /*      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AM-EN
510         26,  3,  0,  2, 15, 35,  1,  3,  3,  0,0,
520         31,  6,  0,  3, 15, 16,  2,  1,  4,  0,0,
530         21,  6,  0,  1, 14, 41,  1,  2,  7,  0,0,
540         31,  7,  0,  3, 15, 20,  0,  2,  7,  0,0}
550 m_vset(5,v)
560   v={          /*       M E L O D Y
570 /*      AF  OM  WF  SY SP  PMD AMD PMS AMS PAN /
580         60, 15,  0,  0,  0,  0,  0,  0,  0,3,  0,
590 /*      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AM-EN
600         31, 12,  1,  1,  0, 22,  0,  2,  3,  0,0,
610         31, 10,  5,  7,  2,  0,  0,  2,  3,  0,0,
620         31, 14,  1,  1,  0, 21,  0,  1,  7,  0,0,
630         31, 10,  6,  7,  2,  0,  0,  1,  7,  0,0}
640 m_vset(6,v)
650   v={          /*       B A C K I N G   P I A N O
660 /*      AF  OM  WF  SY  SP PMD AMD PMS AMS PAN /
670         60, 15,  0,  0,  0,  0,  0,  0,  0,3,  0,
680 /*      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AM-EN
690         31,  0,  0,  0,  0, 34,  0,  2,  3,  0,  0,
700         18, 11,  6,  5,  3,  0,  1,  2,  3,  0,  0,
710         31,  0,  0,  0,  0, 32,  0,  6,  7,  0,  0,
720         31, 11,  7,  5,  3,  2,  1,  2,  7,  0,  0}
730 m_vset(7,v)
740   v={          /*       R I D E   C Y M B A L
                            コノ オトハ 「デンパ」ノ
750 /*      AF  OM  WF  SY  SP PMD AMD PMS AMS PAN /
                            「ネイロ ライブラリー」カラ モッテキマシタ。
760         60, 15,  0,  0,  0,  0,  0,  0,  0,3,  0,
770 /*      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AM-EN
                             スゴク リアル。 ヨク ツクッタナア、コンナ オト
780         31, 31,  0,  0,  1,  9,  0,  3,  0,  1,  0,
790         31, 31,  4,  2,  1, 31,  2,  5,  0,  2,  0,
800         31, 28,  5,  3,  3, 14,  0,  1,  5,  2,  0,
810         31, 31,  5,  2,  7,  4,  2,  7,  0,  3,  0}
820 m_vset(8,v)
830   v={          /*       H A N D   B E L L
840 /*      AF  OM  WF  SY  SP PMD AMD PMS AMS PAN /
850         58, 15, 3,   0,200, 50,  0,  6,  0,3,  0,
860 /*      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AM-EN
870         31, 30,  0,  4,  5, 10,  0, 15,  0,  3,  0,
880         31, 30,  0,  3,  0, 41,  0,  0,  0,  3,  0,
890         31, 27,  0,  4,  0, 17,  0,  0,  0,  0,  0,
900         31, 16, 10,  8, 10,  0,  1, 14,  0,  0,  0}
910 m_vset(9,v)
920   v={          /*       C O W   B E L L
930 /*      AF  OM  WF  SY  SP PMD AMD PMS AMS PAN /
940         59, 15, 0,   0,  0,  0,  0,  0,  0,3,  0,
950 /*      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AM-EN
960         31, 21, 19,  6,  2,  0,  0, 15,  6,  0,  0,
970         31, 21, 12,  6,  2, 35,  0,  8,  6,  1,  0,
980         31, 21, 13,  6,  3, 32,  0,  7,  0,  0,  0,
990         31, 19, 16,  9,  2,  1,  0,  2,  0,  0,  0}
1000 m_vset(10,v)
1010 str p1="y3,1",p2="y3,2",p3="y3,3"
1020 /*
1030 str rd="y2,4",cL="y2,8",th="y2,11",tm="y2,12",tL="y2,13"
1040 str bd="y2,2",ms="y2,10",sd="y2,16",scy="y2,5",kc="y2,23"
1050 str bs="y2,9",cy="y2,3",hc="y2,6",ho="y2,7"
1060 str et1="y2,58",et2="y2,59",et3="y2,60",et4="y2,61"
1070 str tb1="y2,32",tb2="y2,33",tb3="y2,34",tb4="y2,35"
1080 str sd1="y2,16",sd2="y2,15",sd3="y2,14"
1090 /*
1100 key 2,"m_play()"+chr$(13):key 12,"m_stop()"+chr$(13):key 1
4,"save"+chr$(34)+"beyond"
1110 /* m_init():m_trk(5,"t106L16@6v10o5q8p3"+k):m_play()
1120 m_tempo(110)
1130 for i=1 to 7:m_alloc(i,3000):m_assign(i,i):next
1140 m_alloc(8,6000):m_assign(8,8)
1150 str a[256],b[256],c[256],d[256],e[256],f[256],g[256],h[256
],z[256]
1160 str aa[256],bb[256],cc[256],dd[256],ee[256],ff[256],gg[256
],hh[256],zz[256]
1170 str k[256],k1[256],k2[256],k3[256]
1180 str j[256],l[256],m[256],n[256],p[256]
1190 str jj[256],ll[256],mm[256],nn[256],pp[256]
1200 str c1[256],c2[256],d1[256],d2[256],f1[256]
1210 /*
1220 /*        B A S S
1230 /*
1240 a="r2"
1250 b="L16|: e4.rb<ed>bb-age-4.r v10b-rb-rv9 e-b-8 :|
1260 c="d-4.ra-<d->ba-gg-e d4.ra<dc>aa-gf->b4.<rg-bag-fed c4.rg
<c>b-gg-fe-
1270 d="L16|:e1<e2>r8.ef+ga<e e-2.r8>b-8 <e-8.>b-8.e-4.&e-4:|
1280 ee="L16grg8g<g>ggg8g<g>gg<f&g  e-re-8e-<e->e-e-a-b->b-<g-&
a->a-<d-&e- drd8d<d>ddg&add8dd<d >drd8d<d>ddd8{dd<d}8>dd<c&d>
1290 ff=">b-rb-8b-<b->b-rb-8r<b->b-b-<e-&f crc8c<c>ccf&gcc&ddg&
a drd8d<d>ddg&arf&grc&d drd8d<d>ddd8<d8>d8<d8>
1300 g=">L8|:4q8v9g<q4v10g>:||:4q8v9b-q4v10<b->:||:4q8v9g<q4v10
g>:||:4q8v9b-q4v10<b->:|q8v8e1r1
1310 k=">L8|:4q8v9g<q4v10g>:||:4q8v9b-q4v10<b->:||:4q8v9g<q4v10
g>:||:4q8v9b-q4v10<b->:|<
1320 l="L16g8.gr2. r1 g8.grgr8r2 |:r1 g8.grgrgr2:|
1330 n="|: e4r8.<<d&e>aa&bg&ad&e r8.g>ee<d&e>aa&b<dd&eg&a d-&dd
<d>dd<c&d>gg&add&ec&d >g8g<g>gg<f8>gg&a<cc&df&g r>f8.g4rg8.a4 ra
8.b4r2 :|
1340 m_trk(1,"[d.c.]"+a+"@1 [coda]")
1350 m_trk(1,"   o3 q8 v9  p3y48,0")
1360 m_trk(1,b)          :m_trk(1,c)
1370 m_trk(1,b)          :m_trk(1,c)
1380 m_trk(1,"   o2     v9"+d)
1390 m_trk(1,ee)
1400 m_trk(1,ff)
1410 m_trk(1,g)
1420 m_trk(1,"          v8"+d)
1430 m_trk(1,"      o2  v9"+ee)
1440 m_trk(1,ff)
1450 m_trk(1,k)          :m_trk(1,k)
1460 m_trk(1,"      o2  v11"+l)
1470 m_trk(1,"          v10"+n+"[*]")
1480 /*
1490 /*
1500 /*        B A C K I N G and C H O R D
1510 /*
1520 b="L16|: e8g<d>g<gd<c8>gdf>e8e-8g<d>g<gd<c8>gdg>e-8:|
1530 c="d-8faf<e>b<a8e>b<e>d-8 d8faf<fca8fcf>d8 >b8<e-ae-<d>a<g
8d>a<d>>b8 <c8e-b-e-<d>b-<g8d>b-<d>c8
1540 d1="e8g<d>g<g-d>g8<g-dg->e8g<d>":d2="e-8g<d->g<g-d->g8<g-d
-g->e-8g<d->"
1550 dd="L16|:"+d1+":||:"+d2+":||:"+d1+":|"+d2+d2
1560 e="L16 f8.frfrf&f4..r d-1 d8.drdrd&d4..r d8.drdrdd8drd8dr
1570 f=">b-8.b-rb-rb-&b-4..r<c8.crcrc&c4..r d8.drdrdd2 d4&d  v7
a8<c8v6e8.a8.<v4c
1580 g="g1b-1g1b-1 a-1&a-1
1590 k="|:4g1a1:|
1600 l="L16c8.cr2. r1 c8.crcr8r2 r1 c8.crcrcr2 r1  v11 <c8.crcr
cr2
1610 n="L1|:eedd   q7r16c8.d4r16d8.f4 r16f8.g4r2 q8:|
1620 m_trk(2,"[d.c.]"+a+"[coda]")
1630 m_trk(2,"@5 o3 q8 v13 p3y49,0")
1640 m_trk(2,b)          :m_trk(2,c)
1650 m_trk(2,b)          :m_trk(2,c)
1660 m_trk(2,"   o3     v11"+dd)
1670 m_trk(2,"@7 o2 q8 v8  p3 y49,0"+e)
1680 m_trk(2,f)
1690 m_trk(2,"@4 o2     v8  p1"+g)
1700 m_trk(2,"@5 o3     v9     "+dd)
1710 m_trk(2,"@7 o2 q8 v8  p3 y49,0"+e)
1720 m_trk(2,f)
1730 m_trk(2,"@4 o2     v8  p1"+k)
1740 m_trk(2,"@4 o3 q6 v10 p2"+l)
1750 m_trk(2,"   o2 q8 v9  p2"+n+"[*]")
1760 /*
1770 cc=left$(c,len(c)-2)
1780 d="L1|:g-&g-g-&g-:|
1790 e="L16 g8.grgrg&g4..r f1  f8.frfrf&f4..r f8.frfrff8frf8fr
1800 f="f8.frfrf&f4..r          g8.grgrg&g4..r a8.araraa2 a4&a y50
,8 v7a8<c8v6e8.a8.<v4c
1810 g="b1<d1>b1<d1>b1&b1
1820 h=left$(dd,len(dd)-5)
1830 k="|:4b1<d1>:|
1840 l="L16d8.dr2. r1 d8.drdr8r2 r1 d8.drdrdr2 r1  v11 <d8.drdr
dr2
1850 n="L1|:aagg   q7r16f8.g4r16g8.a4 r16a8.b4r2 q8:|
1860 m_trk(3,"[d.c.]"+a+"[coda]")
1870 m_trk(3,"@5 o3 q8 v13 p1y50,20")
1880 m_trk(3,"@w8"+b)          :m_trk(3,cc+"c8")
1890 m_trk(3,b)          :m_trk(3,cc)
1900 m_trk(3,"@4 o3 q8 v8 p1 y50,8"+d)
1910 m_trk(3,"@7 o2 q8 v8 p3 y50,0"+e)
1920 m_trk(3,f)
1930 m_trk(3,"@4 o2     v8 p3"+g)
1940 m_trk(3,"@5 o3     v9 p1 y50,20@w8"+h)
1950 m_trk(3,"@7 o2 q8 v8 p3 y50,0"+e)
1960 m_trk(3,f)
1970 m_trk(3,"@4 o2     v8 p3"+k)
1980 m_trk(3,"@4 o3 q6 v10 p2"+l)
1990 m_trk(3,"   o2 q8 v9  p3"+n+"[*]")
2000 /*
2010 cc=cc+"c"
2020 d="L1|:d&dd-&d-:|
2030 e="L16 b-8.b-rb-rb-&b-4..r g1 <c8.crcrc&c4..r>b8.brbrbb8br
b8br
2040 f="a-8.a-ra-ra-&a-4..r b-8.b-rb-rb-&b-4..r<c8.crcrcc2 c4&c
>y51,12 v7a8<c8v6e8.a8.<v4c
2050 g="d1f1d1f1 d-1&d-1
2060 h=left$(dd,len(dd)-4)
2070 k="|:4d1f1:|
2080 l="L16f8.fr2. r1 f8.frfr8r2 r1 f8.frfrfr2 r1  v11 <f8.frfr
fr2
2090 n="L1|:dd-c>b q7r16a8.b4r16b8.<c4 r16c8.d4r2 q8:|
2100 m_trk(4,"[d.c.]"+a+"[coda]")
```

```
 2110 m_trk(4,"@5 o3 q8 v13 p2y51,28")
 2120 m_trk(4,"@w16"+b)          :m_trk(4,cc+"8")
 2130 m_trk(4,b)          :m_trk(4,cc)
 2140 m_trk(4,"@6 o4 q8 v8  p2y51,4"+d)
 2150 m_trk(4,"@7 o2 q8 v8  p3 y51,0"+e)
 2160 m_trk(4,f)
 2170 m_trk(4,"@4 o3     v8 p3"+g)
 2180 m_trk(4,"@5 o3     v9 p2 y51,20@w16"+h)
 2190 m_trk(4,"@7 o2 q8 v8  p3 y51,0"+e)
 2200 m_trk(4,f)
 2210 m_trk(4,"@4 o3     v8 p3"+k)
 2220 m_trk(4,"@4 o3 q6 v10 p1"+l)
 2230 m_trk(4,"   o3 q8 v9  p3"+n+"[*]")
 2240 /*
 2250 /*         M E L O D Y
 2260 /*
 2270 a="r2"
 2280 /*
 2290 d="L16|:g-2.d8eg-&g-4.e8g-8.g8.a8g-1& |1g-4r2>b<d-de:|g-1
 2300 e="L16f64&f+64&g2.&g.  fef2.&f8ef e2d64&e-64&e32&e8.&d8c8>
 b8<cd2&d>gab-<c
 2310 z=S("g+",6,252,0,5):zz=S("g",6,252,0,5)
 2320 f="d-4.c8d-8e-fr4 e-8fg8f32&g-32&g8g-32&g32&g8a-8.a8 a1 a-
32&a.&a& @L1"+z+"&"+zz+"&y52,0g4& v9g4& v8g4
 2330 g="L16g2..fe fed4.c8.d8.e8 d32&e-32&e8.&e8dc>b8.<c8.d8> a4
8&b-48&b48&bg4gfefgab<cdf e1&e2r8
 2340 gg="rc8cc8
 2350 z=S("e",7,252,0,5):zz=S("d+",7,252,0,5)
 2360 k="L16r8c32&d-32&d4.&d4c8>b-a-64&a..b-<c8>a-64&a..e32&f.&@
L6"+z+"&"+zz+"L16y52,0c+32&d32&d2.<d4c8>a+8ab<c8>a8fg8& @L4
 2370 z=S("f+",18,252,0,5)
 2380 k1=z+"y52,0L16rg<cdd8cd8c8d8f64&f+64&g8..&g2fe-8d8& @L2
 2390 z=S("c+",12,252,0,5):zz=S("c+",12,0,252,5)
 2400 k2=z+"&y52,0c+8&"+zz+"&y52,0d2 L16
 2410 z=S("c",6,0,252,5):zz=S("c",6,252,0,5)
 2420 k3=">>{fa<cd+f+a}4>{a<cd+f+a<c}4>{f+a<cd+f+a}4 b< @L2"+z+"
&y52,0 L16dc+32& @L1"+zz+"&"
 2430 l=">y52,0b8r8" : ll="L4|:27c:|
 2440 n="L8|:12ccc16cc.c.c.:|
 2450 m_trk(5,"[d.c.]"+a+"[coda]")
 2460 m_trk(5,"@7 o2 q8 v5  p3y52,12")
 2470 m_trk(5,b)          :m_trk(5,c)
 2480 m_trk(5,b)          :m_trk(5,c)
 2490 m_trk(5,"@6 o4 q8 v10 y52,0"+d)
 2500 m_trk(5,"          v11"+e)
 2510 m_trk(5,f)
 2520 m_trk(5,"   o4     v10y52,0 "+g)   :m_trk(5,"@8 o5 v15"+gg)
 2530 m_trk(5,"@5 o3     v8 y52,20"+dd)
 2540 m_trk(5,"@6 o4     v11"+e)
 2550 m_trk(5,f)
 2560 m_trk(5,"@6 o5     v11   y52,0"+k) :m_trk(5,k1)    :m_trk(5
,k2)    :m_trk(5,k3)
 2570 m_trk(5,l)          :m_trk(5,"@10 o4 v15 y52,0"+ll)
 2580 m_trk(5,n+"[*]")
 2590 /*
 2600 /*         M E L O D Y   E C H O   P A R T
 2610 /*
 2620 b="|:4r2r4.:|":c=b
 2630 gg="c8ccc8
 2640 h="|:r1r1r1r1:|"
 2650 z=S("e",7,252,0,6):zz=S("d+",7,252,0,6)
 2660 k="r4L16r8c32&d-32&d4.&d4c8>b-a-64&a..b-<c8>a-64&a..e32&f.
&@L6"+z+"&"+zz+"L16y53,12c+32&d32&d2.<d4c8>a+8ab<c8>a8fg8& @L4
 2670 z=S("f+",18,252,0,6)
 2680 k1=z+"y53,12L16rg<cdd8cd8c8d8f64&f+64&g8..&g2fe-8d8& @L2
 2690 z=S("c+",12,252,0,6):zz=S("c+",12,0,252,6)
 2700 k2=z+"&y53,12c+8&"+zz+"&y53,12d2L16
 2710 z=S("c",6,0,252,6):zz=S("c",6,252,0,6)
 2720 k3=">>{fa<cd+f+a}4>{a<cd+f+a<c}4>{f+a<cd+f+a}4
 2730 l="L16g8.gr2. r1 g8.grgr8r2 r1 g8.grgrgr2 r1  v11 <g8.grgr
gr2
 2740 n="L16|:12g8<g>g8g8g<p1g8p2>g<g8>g<g8>:|
 2750 m_trk(6,"[d.c.]"+a+"[coda]")
 2760 m_trk(6,b)          :m_trk(6,c)
 2770 m_trk(6,b)          :m_trk(6,c)
 2780 m_trk(6,"@6 o4 q8 v9 y53,12"+d)
 2790 m_trk(6,"@1 o2 q8 v9 y53,8"+ee)
 2800 m_trk(6,ff)
 2810 m_trk(6,"@6 o4     v9 y53,12"+g)   :m_trk(6,"@3 o3 v12"+gg)
 2820 m_trk(6,h)
 2830 m_trk(6,"@1 o2 q8 v9 y53,8"+ee)
 2840 m_trk(6,ff)
 2850 m_trk(6,"@6 o5     v7     y53,12"+k) :m_trk(6,k1)    :m_trk(
6,k2)    :m_trk(6,k3)
 2860 m_trk(6,"@4 o3 q6 v9     y53,20"+l)
 2870 m_trk(6,"@5 o5 q8 v14 p2y53,0"+n+"[*]")
 2880 /*
 2890 /*         E T C   P A R T   ( ? ? )
 2900 /*
 2910 b="L16|:4r2r4.:|"
 2920 c="|:@3c8c8c"+p1+ho+p3+"@2c@3c8c8c8c"+p1+ho+p3+"@2c:|@3c"+
p1+ho+p3+"@2c@3c8c"+p1+ho+p3+"@2c@3c"+p1+ho+p3+"@2c@3c8c8c"+p1+h
o+p3+"@2c@3c8c8c"+p1+ho+p3+"@2c@3c8c8c8c8
 2930 d="L1|:d&dd-&d-:|
 2940 e="L16 d8.drdrd&d4..r d-1 e8.erere&e4..r e8.ereree8dre8dr
 2950 f="d-8.d-rd-rd-&d-4..r  e-8.e-re-re-&e-4..r e8.ereree2 e4&
e>y54,12 v7a8<c8v6e8.a8.<v4c
 2960 g="g1a1g1a1 e1&e1
 2970 h="L16c8ccrccrc8ccrccr ccrccrc8c8ccc8c8 c8ccrccrc8ccrccr c
crccrc8c8ccc8c8
 2980 k="|:4g1f1:|"
 2990 l="L16g8.gr2. r1 g8.grgr8r2 r1 g8.grgrgr2 r1  v11 <g8.grgr
gr2
 3000 n="L1|:ggff>  q7r16f8.g4r16g8.a4 r16a8.b4r2 q8:|
 3010 m_trk(7,"[d.c.]"+a+"[coda]")
 3020 m_trk(7,"o5 v12 q8")
 3030 m_trk(7,b)          :m_trk(7,c)
 3040 m_trk(7,b)          :m_trk(7,c)
 3050 m_trk(7,"@4 o3 q8 v8  p3 y54,20"+d)
 3060 m_trk(7,"@7 o3 q8 v8  p3 y54,0"+e)
 3070 m_trk(7,f)
 3080 m_trk(7,"@4 o3 v8     p2"+g)
 3090 m_trk(7,"@8 o5 v15"+h)     :m_trk(7,h)
 3100 m_trk(7,"@7 o3 q8 v8  p3 y54,0"+e)
 3110 m_trk(7,f)
 3120 m_trk(7,"@4 o3     v8  p2"+k)
 3130 m_trk(7,"@4 o3 q6 v10 p1"+l)
 3140 m_trk(7,"   o3 q8 v9  p1"+n+"[*]")
 3150 /*
 3160 /*         D R U M S
 3170 /*
 3180 a="L16o5v12q8"+p3+bd+"r"+bd+"r"+bs+"r"+ms+"r"+sd+"r"+p2+tm
+"r"+p3+tm+"r"+p1+tL+"r"+p3
 3190 b="L16|:3"+bd+"@3c"+p1+ho+"r@3c&"+p3+bd+"c"+sd+"c"+p1+ho+"
r@3c&"+p3+bd+"cc8"+sd+"c8c8:|"
 3200 b=b+bd+"c"+p1+ho+"rc&"+p3+bd+"c"+bs+"c@2v14|:"+ms+"c32&"+m
s+"c32@3v12|:"+sd+"c32&:|"+sd+"rc"+sd+"r"+p2+th+"c"+p3+tm+"r"+p1
+tL+"c"+p3+bd+"r"
 3210 c=bd+"r8."+bd+"r"+sd+"r8|:"+bs+"r:||:"+ms+"r:|"+p2+"|:"+th
+"r:|"+p3+tm+"r"+p1+tL+"r"+p3
 3220 c=c+bd+"r8"+bd+"r8 @L8"+sd+"r"+sd+"rr r"+p2+th+"r"+th+"r "
+p3+tm+"r"+tm+"r"+p1+tL+"r"+tL+"r"+tL+"r"+p3+sd+"r L16r8"
 3230 c=c+bd+"r8."+bs+"r"+ms+"r"+sd+"r8"+sd+"r"+sd+"r8"+bs+"r"+m
s+"r"+sd+"r"+sd+"r"
 3240 c1=bd+"r8."+bd+"r"+sd+"r"+bd+"r|:4"+bs+"r:||:4"+sd+"r:|"
 3250 c2="@L8 |:3"+p2+th+"r"+th+"r "+p3+tm+"r"+tm+"r"+p1+tL+"r"+
tL+"r:|"+p3+sd+"r8"
 3260 d="L16@3v10|:3"+bd+"ccc"+bd+"c"+sd+"c"+bd+"cc"+bd+"cccc"+b
d+"c"+sd+"cccc:|"+bd+"ccc"+bd+"c"+sd+"c"+bd+"cc"+bd+"c"+sd+"c"+s
d+"c"+p2+"|:"+th+"c:|"+p3+"|:"+tm+"c:|"+p1+tL+"c"+ho+"r"+p3
 3270 e="|:3@3v10"+bd+"cc"+p1+ho+p3+"r@2c"+sd+"@3cc"+p1+ho+p3+"r
@2c"+bd+"@3cc"+p1+ho+p3+"r"+bd+"@2c"+sd+"@3cc"+p1+ho+p3+"r@2c:|"
+p3
 3280 ee=bd+"cc"+p1+ho+p3+"r@2c"+sd+"@3cc"+p1+ho+p3+"r"+sd+"@2c"
+bs+"c"+ms+"c"+ms+"@2c&"+sd+"c"+p2+th+"@3c"+p3+sd+"c"+tm+"@2c&"+
p1+tL+"c"+p3
 3290 g="@3|:3"+bd+"cc"+p1+ho+p3+"rr"+sd+"cc"+p1+ho+p3+"rr"+bd+"
cc"+p1+ho+p3+"rr"+sd+"cc"+p1+ho+p3+"rr:|"
 3300 g=g+bd+"cc"+p1+ho+p3+"rr"+sd+"cc"+p1+ho+p3+"rr"+bd+"@L8c.@
L4c&"+bs+"@L8c"+bs+"@2c&"+ms+"c&"+ms+"c L16"+sd+"@3c"+p2+th+"c"+
p3+tm+"@2c&"+p1+tL+"c"+p3
 3310 gg=p1+hc+"@8v15o5c8"+p1+hc+"c"+hc+"cr"+hc+"c"+hc+"cr"+ho+"
c8r"+hc+"c8"+p3+sd+"c"+p1+hc+"c8"+hc+"c8"+hc+"c"+hc+"cr"+hc+"c"+
hc+"cr"+ho+"c8 @L8|:3"+p3+bs+"r:||:3"+ms+"r:||:3"+sd+"r:|"+p3
 3320 h="L16 @3v11o5"+bd+"c8c"+sd+"cc8c8c8c8"+bs+"c"+ms+"c"+sd+"
c&"+bd+"c"+bd+"c8c8c8c8"+p1+ho+p3+"r8c8"+p1+ho+p3+"r"+ms+"c"+sd+
"c8"
 3330 hh=bd+"c8c8c8c8"+p2+cL+"c&"+cL+"cc8"+p3+bs+"c&"+bs+"c"+ms+
"c&"+sd+"c L32"+bd+"c&"+bd+"c&"+bd+"c&"+bs+"c|:3"+bs+"c&:|"+ms+"
c L16"+ms+"c&"+sd+"c"+bd+"c8"+p1+ho+"@9v12p1>aa@3v11<c8"+ho+p3+"
r"+bs+"c"+bs+"c8"
 3340 j=bd+"c8c"+ms+"cc8c8c8c8"+bs+"c"+ms+"c"+sd+"c&"+bd+"c"+bd+
"c8c8 @9v12o4aa @3v11o5c8"+p1+ho+p3+"r8c8"+p1+ho+p3+"r"+ms+"c"+s
d+"c8"
 3350 jj=bd+"c8c8c8c8"+p2+cL+"c&"+cL+"cc8"+p3+bs+"c&"+bs+"c"+ms+
"c&"+sd+"c"+bd+"c8c&"+bd+"c"+sd+"c&"+bd+"cc&"+bd+"c"+sd+"@2o5v11
c&"+sd+"c@3"+p2+th+"c&"+th+"c"+p3+tm+"@2c&"+tm+"c"+p1+tL+"c8"+p3
 3360 l=bd+"@10o4v15c8."+bd+"@2o5v12c@3c"+bs+"c"+ms+"c"+bd+"cc"+
bd+"@2c@3cc"+bd+"c"+sd+"ccc"+bd+"c"+bs+"c"+p1+ho+"rc"+p3+bd+"L32
|:6c:|L16"+sd+"c"+bd+"c"+p2+th+"cc"+p3+tm+"c"+p1+tL+p3+"c"+bd+"c
@9o4|:"+bd+"a:|"
 3370 ll=bd+"@2o5c8."+bd+"c@3c"+bd+"@2c@3@L4c&cc&"+bd+"cc&c"+bs+
"c&c&"+bs+"cc&"+bs+"c&c"+ms+"c&c&"+ms+"cc&"+sd+"c&cL16c|:3"+bd+"
c:|c"+bd+"cc"+bd+"cc"+bs+"c"+ms+"cc"+bd+"cc"+bd+"cc"+bs+"c"+p1+h
o+p3+"r8."
 3380 m=bd+"@2c8."+bd+"c@3c"+bd+"@2c@3c"+bd+"@2c @3 @L8"+p1+tb2+
"c&c&"+p3+tb1+"cc&"+tb1+"c&c"+tb1+"c&c&"+p2+th+"cc&"+p3+tm+"c&c"
 3390 mm=p1+tL+p3+"L16c&"+bs+"c"+bs+"c&"+ms+"c"+ms+"c&"+bd+"cc&c
"+sd+"c"+bd+"@2c @3 L32cccc"+bd+"ccL16"+sd+"cr"+bd+"r"+bd+"@2c8.
"+bd+"c@3c"+bd+"@2c@3c"+bd+"@2c@3crr"+bd+"r"+sd+"c4"
 3400 n="L16v10|:4"+bd+"cccc"+sd+"ccc"+bd+"cc"+bd+"c"+bd+"cc"+sd
+"cccc:|"
 3410 n=n+"c"+bd+"@2c@3cc"+bd+"ccc"+sd+"cc"+bd+"@2c@3cc"+bd+"ccc
"+sd+"c c"+bd+"@2c@3cc"+bd+"ccc"+sd+"cc"+sd+"c"+bd+"cc @L4"+bs+"
c&c&cc&"+ms+"c&cc&c&"+ms+"cc&c&c"
 3420 m_trk(8,"[d.c.]"+a+"[coda]")
 3430 m_trk(8,b)          :m_trk(8,c):m_trk(8,c1)
 3440 m_trk(8,b)          :m_trk(8,c):m_trk(8,c2)
 3450 m_trk(8,d)          :m_trk(8,d)
 3460 m_trk(8,e)          :m_trk(8,ee)
 3470 m_trk(8,e)          :m_trk(8,ee)
 3480 m_trk(8,g)          :m_trk(8,gg)
 3490 m_trk(8,h)          :m_trk(8,hh)
 3500 m_trk(8,j)          :m_trk(8,jj)
 3510 m_trk(8,e)          :m_trk(8,ee)
 3520 m_trk(8,e)          :m_trk(8,ee)
 3530 m_trk(8,g)          :m_trk(8,g)
 3540 m_trk(8,l)          :m_trk(8,ll)
 3550 m_trk(8,m)          :m_trk(8,mm)
 3560 m_trk(8,n)          :m_trk(8,n+"[*]")
 3570 m_play()  :end
 3580 /* E A S Y   B E N D   R O U T I N E
 3590 /* A="C".."G" , L=length , V1=start KF(0-252) , V2=end KF(
0-252) , ch=channel(1-8)
 3600 func str S(A;str,L;float,V1;float,V2;float,ch;char)
 3610 str B[256]
 3620 float VL,V
 3630 VL=(V2-V1)/(L-1):B="":V=V1
 3640 for I=1 to L:if V>252 then V=252 else if V<0 then V=0
 3650 B=B+"y"+str$(47+ch)+","+str$(int(V))+A:V=V+VL
 3660 if I<>L then B=B+"&"
 3670 next
 3680 return(B)
 3690 endfunc
```

マシン語カクテル in Z80's Bar

第6回——東京3D迷路物語——

シナリオ/イラスト：山田純二
特別監修：浦川博之　金子俊一

Z80's Barはプログラマの憩いの場。というわけで，今夜はMZ-2200ユーザーの山田純二くんがやってきました。なんでも3D迷路の表示で悩んでいるとか。さあ，今宵も長老の講釈が冴えわたるか？ おや，いつになくようこさんの様子がおかしいぞ……。

♪カランコローン（ドアベルの音）

ようこ（以下Yo）：いらっしゃいませ。（ぶすーとした声）

マスター（以下M）：あ，長老今晩は，今日はずいぶんと懐かしい人がきてますよ。

山田純二（以下純）：今晩は，長老。

長老（以下老）：おお，これはこれは山田君ではないか。元気にしとったかのう。

純：はい，テトリスだったらまかせてください。

老：そっそういうことを聞いたわけではないんだが……。そういえば君はMZを使ってたんではなかったかのう。いまでも使っとるのかな？

純：もちろんですよ，愛機ですから。最近ではマシン語の勉強も始めまして，毎日キーボードをたたいてます。この間なんか思いっきりリターンキーたたいたら，すぱこ～んと飛んじゃいましたけどね。

老：そうかそうか元気でなによりじゃ。しかし愛機だったらもっと丁寧に扱わんといかんぞ。

純：はーい。

Yo：（つかつかと寄ってきて）何になさいます。

老：お，ようこちゃんどうしたのじゃ，そんな怖い顔をして。

Yo：アルファエー（注1）でよろしいですか。

老：うっ，まあそれでよい。マスターどうしたのじゃ，ずいぶんと今日はようこちゃんの機嫌が悪いようじゃが。

M：それはですね，今日は光君，て君，善君とメアリーが一緒に，友達の水野君がバイトしている東京ディズニーランドへ遊びにいっているんですよ。

老：おお～，あの水野君か。最近見かけんと思っとったがデゼニランド（注2）でバイトとはのう。なるほど，あやつらが居らんとここも静かなもんだ。

光&て&善&メ：はっくしょん！

M：そこで問題なんですが，あのメアリーがずいぶん光君のことを気に入ったらしく，べったりくっついてたりするわけですよ。

老：そうかそうか，いわゆる"ちぇるしい"というやつじゃな。

純：アナタニモチェルシーアゲターイ。じゃなくって，それをいうならジェラシーでしょ長老。

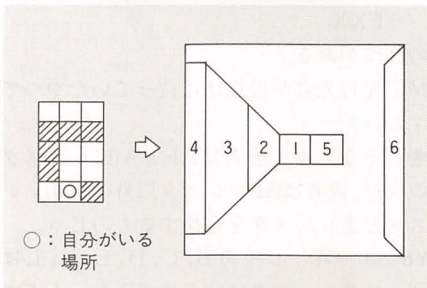
老：そうとも言うかのう，最近は。それより純二君，いきなり訪ねてきたりして，さてはよほど毒物飲料が恋しくなったとか？

純：そんなわけありませんよ。実はいま，3DダンジョンタイプのRPGを作ろうとしているのですが，肝心の3D迷路の表示のやり方がわからなくて，最初から行き詰まってしまったんです。

老：ほほう，3D迷路か，いまではこれぐらいのアルゴリズムは知っておらんと話にならんからのう。よしっ，このわしにまかせなさい。

純：よろしくお願いします。

迷路を表示するアルゴリズム

老：ではと，静かなところで始めるとするか。マスター，以前わしがMZで作ったプログラムがあったはずじゃが……，見つかるかのう。

M：探してみましょう。

老：よろしく頼むよ。ではプログラムが見つかるまでアルゴリズムの説明でもしていよう。時に純二君，迷路のデータはどういう形式にしようとしておるのじゃ。

純：えーと，"0"がなにもないときで，"1"のときは壁があることにして，それ以外の数はなにかイベントが起きるようにしようと思っています。

老：ふむ，いちばん基本的な形式じゃな。では，図1のような場合を考えてみよう。表示の手順は，

1) まず，真ん中を手前から奥に向かって，表示範囲内に壁があるかどうか調べていく。
2) 壁があったら表示する。そして，そこの位置から左右それぞれの壁を奥から手前に向かって表示していく。壁がなかったら，いちばん奥からそれぞれ左右の壁を表示する。

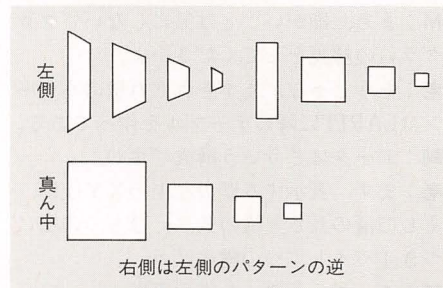
と，これだけじゃ。

純：へー，あきれるくらい簡単ですね。あっ，でも長老，壁は奥へ行くほど小さく表示されているわけですが，これはどうやっているんですか？

老：それはじゃな，図2のようにあらかじめパターンを用意しておくのじゃ。

純：なるほど，あとは壁があった場合とな

図1　3D迷路で壁を表示する順番

図2　あらかじめ用意しておくパターン

かった場合，それぞれのパターンを表示してやればいいんですね。

老：まあだいたいそんなところじゃな。しかし，それだけでは正しく表示されない場合が出てきてしまうのじゃ。どういう場合かわかるかの？

純：えー，そんなこと言われても，あっ，ようこさんでしたっけ？　……わかりますか？

Yo：わかるわけないでしょ。

老：こらこら，すぐに人に頼ろうとしおって，少しは自分で考えたらどうじゃ。

純：すんまへん。

老：まあよい。たとえば，空白が２コマ以上続いた場合がそうじゃ。このとき，なにも表示されないはずの場所に壁が表示されてしまうのじゃ。

純：あっそうか，つまりひとつ後ろが空白のときに壁を表示しちゃいけないということですか。

老：そのとおりじゃ。時にマスター，プログラムは見つかったかのう。

M：はーい，いま持って行きますよっと，これでしょ，長老。

老：そうじゃ，そうじゃ，ずいぶんと懐かしいのう。

純：江戸時代の話ですからねー。

老：こらこら，わしゃそんな年寄りではないぞ！　だいいちコンピュータ自体，そんな時代には存在しておらんわい。

純：そーでした，そーでした。

老：まったく失礼じゃのう。

M：まあ，細かいことは気にしないでプログラムの解説をしてくださいよ。

老：そうじゃな，まずそれぞれ壁のパターンはLARIT以降にテーブルを作ってある。

純：データはどういう構成ですか。

老：まず，表示する壁の左上のＸＹ座標，そして縦の長さと横の長さ，ひとつの壁につき４バイトずつの構成じゃ。

純：で，さっきやった表示手順１），２）に従って迷路を表示していくわけですね。

老：そうじゃ，MIDKABE，LEFTKABE，そしてRIGHTKABEの順に壁を書いていくのじゃ。

M：ねえ長老，KPRTSUB中の，

```
EX   AF,AF'
```

はなにをやっているんですか？

老：これは，ひとつ後ろになにがあったかA'レジスタに覚えておくためのものじゃ。迷路になにもなかった場合には，このA'レジスタを調べて，壁を表示するかしないかの判定をするのじゃ。

純：なるほど，空白が２コマ以上続いた場合をチェックするためですね。

老：そうして仮想画面に描かれた迷路を最後にテキスト画面に表示しておしまいじゃ。

純：えっ，仮想画面!?

M：ああ，先月西川君がやったあれですね。でも長老，今回はべつにスクロールをさせているわけでもないのにどうして仮想画面を使うんですか？

老：それはじゃな，直接画面を消してから表示をする，これを連続して行うと画面がちらついてしまうのじゃ。しかし，仮想画面を用いれば画面に表示だけを行えばいいので，ちらつきが少なくなるのじゃ。

純：ふーん。

スタック＆交換命令

老：一応３Ｄ迷路の説明が終わったわけじゃが，まだ時間もあることだし，なにか教えてほしいことはないか？

Yo：それじゃ，長老さん，交換命令とスタック操作について教えてほしいな。

老：おっ，ようこちゃんか。もしかして学校の宿題とかいうやつかな？

Yo：そうなの，本当は光君に聞こうと思ったけど今日はいないし……。それからさっきはごめんなさい。

老：べつに気にしておらんよ。ではようこちゃんのリクエストにこたえて交換命令から始めるとするか。まず表と裏レジスタを交換するための命令には，

```
EX   AF,AF'
EXX
```

の２つがある。

M：先月光君が得意げに言っていたやつですね。

老：そうじゃ，前者はAFとＡＦ'レジスタの交換，後者はAFレジスタ以外の汎用レジスタと裏レジスタを交換するものじゃ。

Yo：ふーん，じゃあB，C，D，E，H，Lは別々に裏レジスタには交換できないんですね。

老：残念ながらできないのじゃ。だから，ほかのレジスタを壊さずにＢＣレジスタの値をH'L'に引き渡したいときには，

```
PUSH  BC
EXX
POP   HL
EXX
```

とするしかない。

純：ほかにはどんなものがありますか？

老：あとはHLレジスタとDEレジスタを交換する，

```
EX    DE,HL
```

と，HL，IX，IYの各内容とスタックの内容を交換する，

```
EX    (SP),HL
EX    (SP),IX
EX    (SP),IY
```

などがある。特に「EX　DE,HL」は１命令で双方のレジスタを高速に入れ替えることができるので非常に便利なものじゃ。

Yo：それじゃ，あとの３命令はどういった場合に使われるの？

老：たとえば，一度スタックに積まれた値を変更したいときや，CALL命令でサブルーチンコールされた以外の番地に戻りたいときなどじゃ。

M：いわゆるRETURN［行番号］ですね。

老：そのとおり。しかしあまり使いすぎるとプログラムが汚くなってわけがわからなくなるので注意しなければならない。初心者にはあまり使ってほしくないテクニックのひとつじゃ。

純：怖いですねー。

老：次にスタック操作じゃが，いままでやったPUSH，POP，CALL命令を使うといろいろ面白いことができるのじゃ。

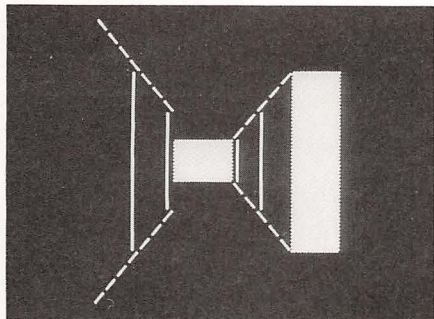
Yo：どういったことができるんですか？

老：たとえば，スタックを使って「JP BC」などをやりたいときは，疑似的に，

```
PUSH  BC
RET
```

で行うことができる。そして，

```
LB1:   CALL  COM
       LD    HL,LB1
       PUSH  HL
       CP    "A"
       JP    Z,ACOM
       CP    "B"
       JP    Z,BCOM
       :
       RET
```

とした場合，「PUSH HL」以降のJP命令はCALL命令として扱われるのじゃ。

Yo：どーして，どーして？

老：CALL命令というのは，どういう動作をするのか知っているじゃろ。

純：確かCALL命令は戻り番地をスタックにPUSHしてからジャンプするんですよね。すると……，

Yo：そうか，JP命令は，LB1を戻り番地とするCALL命令になるんだ。

老：よくわかったのう。このテクニックを使うと，コマンド判定などで条件分岐による大量のサブルーチンコールがあるときに，非常にすっきりまとまるし，リストも短くできるのじゃ。

純：ほえーわけわからん。脳ミソがモヤシになっちゃう！

Yo：飲み物のせいよ，きっと。

老：まあそうじゃろう。いきなり初心者が理解できるものじゃないからな。初心者はとりあえず目をつぶっておいたほうがいいじゃろう。おっともうこんな時間じゃな，それでは皆の衆元気でな。ウハハハハ……。

純：黄金バットみたい。

M：つかれましたね。

Yo：ふーん，マシン語って奥が深いのね。

純：それじゃ私もこのへんで。

M：あっ純二君，今日の分はつけときますからゲームが完成したときに，ちゃんと勘定払ってくださいよ。

純：ほえーい。

—つづく—

注１：アルファルファ（よーするに青くさいモヤシみたいなやつ）のエキスを抽出した飲み物。ただひたすらにナマぐさい。

注２：埼玉にある……わけないか。その昔，ハドソンソフトから出ていた（当時の）名作ソフト。コマンド入力形式のアドベンチャーゲームである。

＊　　＊　　＊

お詫び：先月号の解説でわかりにくい表現がありました。116ページ１段12行目に「M：１行のいちばん左から右へ左に向かって……」とありますが，「M：１行のいちばん右から“左→右”と左に向かって……」というように変更してください。また，同じページの３段18行目の（テキスト）と（アトリビュート）が逆になっていました。申し訳ありません。

MASTER'S MEMO

○Z80におけるレジスタは，普段使っている表レジスタと，直接さわることができない裏レジスタという2種類のレジスタに分類できる（R，Ｉレジスタは特別なのでここでは考えないことにする)。この表と裏のレジスタはその名が示すとおり表裏一体となっている。なぜ裏レジスタが直接さわれないかというと，たとえば紙を机の上に置いてものを書く場合を考えればわかりやすいだろう。裏側に書きたいときは直接書けないから紙をひっくりかえして書くように，レジスタも裏レジスタはひっくりかえす命令を使ってからさわらなければいけない。

○紙で考えて，裏の裏が表であるように，CPUにおけるレジスタもどっちが表で……なんてことは決まっていない。自分で決めてよいのだが，逆にいえば裏表がわからなくなったときはバグにつながるのが目に見えている。初心者が裏レジスタに手を出すのは遠慮したほうがいいだろう。

○IX，IYレジスタには裏レジスタがない。Z80には隠れテクがわりとあるので，IX′，IY′もあると思っていたのだが，確かめてみたらなかったのだ。

○紙をひっくりかえす命令は

EX　AF,AF′

EXX

がある。わざわざ2つに分けた理由としてはフラグレジスタをBC′などと使いたかったからだと思う。そういえば中級者程度でもよくやるミスなのだが，「EX　AF,AF′」ではFレジスタもひっくりかえることに注意が必要。Fレジスタが変わるのだから，このあとに条件式（JP　C，など）を入れたらかわいそうな結果だけが待っている。

○「EX DE,HL」は，単にレジスタペアの中身だけを交換する命令。

○Z80ではレジスタに（　）をつけると，そのレジスタがさすアドレスの中身を表す。たとえば，HLに8000_H，8000_H番地には84_Hが入っていたとすると，HLは8000_H，（HL）は84_Hを表すことになる。

○EX　(SP),HL

EX　(SP),IX

EX　(SP),IY

この3つの命令はスタックを直接操作することになるので，スタック操作に慣れるまでは使わない方が賢明であろう。

一応動作例を示すと，

SP　＝　$F000_H$

($F000_H$)　＝　(SP)　＝　12_H

($F001_H$)　＝　(SP+1)　＝　34_H

H　＝　56_H

L　＝　78_H

とすると，「EX　(SP),HL」で，

SP　＝　$F000_H$

($F000_H$)　＝　(SP)　＝　78_H

($F001_H$)　＝　(SP+1)　＝　56_H

H　＝　34_H

L　＝　12_H

になる。よ〜く見比べるように。

リスト１　3D　MAZEルーチン

```
 1
 2 ;
 3 ; MZ-2200 3D MAZE
 4 ;     in Z80's Bar  BY Chourou
 5
 6          ORG   $8000
 7 BAFA     EQU   $9000          ;ｶｿｳｶﾞﾒﾝ ｱﾄﾞﾚｽ
 8
 9 MAIN
10          LD    A,06
11          CALL  $08C6          ;CLS
12          CALL  BAFACLS
13          CALL  MIDKABE
14          CALL  LEFTKABE
15          CALL  RIGHTKABE
16          CALL  TENSOU
17          RET
18
19 ;ﾋﾀﾞﾘ ﾉ ｶﾍﾞ PRINT
20 LEFTKABE
21          LD    HL,LARIT
22          LD    (PT2+2),HL
23          LD    HL,LNAIT
24          LD    (PT4+2),HL
25          LD    HL,LARIKABE
26          LD    (PT3+1),HL
27          LD    HL,MAZEDAT+4
28          CALL  KABEPRINTSUB
29          RET
30
31 ;ﾐｷﾞ ﾉ ｶﾍﾞ PRINT
32 RIGHTKABE
33          LD    HL,RARIT
34          LD    (PT2+2),HL
35          LD    HL,RNAIT
36          LD    (PT4+2),HL
37          LD    HL,RARIKABE
38          LD    (PT3+1),HL
39          LD    HL,MAZEDAT+14
40          CALL  KABEPRINTSUB
41          RET
42
43 KABEPRINTSUB
44          LD    A,(OKUYUKI)
45          LD    B,A
46          LD    A,05
47          SUB   B
48 LKB8
49          DEC   A
50          JR    Z,LKB9
51          DEC   HL
52          JR    LKB8
53 LKB9
54          LD    A,(HL)
55          EX    AF,AF'
56          DEC   HL
57          LD    A,(OKUYUKI)
58          LD    B,A
59 LKB2
60          PUSH  BC
61          LD    A,(HL)
62          OR    A
63          JR    Z,LKB4
64
```

```
 65          EX    AF,AF'
 66 PT2      LD    IX,LARIT
 67          DEC   B
 68          JR    Z,LKB3
 69          CALL  IXADRSET
 70 LKB3
 71          PUSH  HL
 72 PT3      CALL  LARIKABE
 73 LKB5
 74          POP   HL
 75 LKB7
 76          DEC   HL
 77          POP   BC
 78          DJNZ  LKB2
 79          RET
 80 LKB4
 81          EX    AF,AF'
 82          OR    A
 83          JR    Z,LKB7
 84 PT4      LD    IX,LNAIT
 85          DEC   B
 86          JR    Z,LKB6
 87          CALL  IXADRSET
 88 LKB6
 89          PUSH  HL
 90          CALL  NAIKABE
 91          JR    LKB5
 92
 93 ;ﾏﾝﾅｶ ﾉ ｶﾍﾞ PRINT
 94 MIDKABE
 95          LD    HL,MAZEDAT+6
 96          LD    B,04
 97          LD    C,02
 98 MID2
 99          LD    A,(HL)
100          OR    A
101          JR    NZ,MID3
102          INC   HL
103          INC   C
104          DJNZ  MID2
105 MID7
106          LD    A,4
107          LD    (OKUYUKI),A
108          RET
109 MID3
110          DEC   C
111          LD    A,C
112          LD    (OKUYUKI),A
113          LD    IX,MIDTABLE
114 MID4
115          DEC   B
116          JR    Z,MID5
117          INC   IX
118          INC   IX
119          JR    MID4
120 MID5
121          LD    B,(IX+00)
122          LD    C,B
123          CALL  ADRSET
124          LD    B,(IX+01)
125          LD    C,B
126          CALL  NAIKABESUB
127          RET
128 IXADRSET
129          INC   IX
130          INC   IX
131          INC   IX
132          INC   IX
133          DJNZ  IXADRSET
134          RET
135
136 ;ｶﾍﾞ ﾉ PRINT  IN IX-ﾃｰﾌﾞﾙ ｱﾄﾞﾚｽ
137 NAIKABE
138          CALL  PARASET
139          CALL  NAIKABESUB
140          RET
141
142 NAIKABESUB
143          PUSH  BC
144          PUSH  HL
145          LD    A,$1F
146          CALL  LINE
147          POP   HL
148          INC   HL
149          POP   BC
150          DEC   C
151          JR    NZ,NAIKABESUB
152          RET
153
154 LARIKABE
155          CALL  PARASET
156          DEC   C
157          JR    Z,LAK3
158 LAK2
159          LD    A,$5C
160          LD    (HL),A
161          LD    DE,21
162          ADD   HL,DE
163
164          PUSH  HL
165          PUSH  BC
166          LD    A,$20
167          CALL  LINE
168          LD    A,$2F
169          LD    (HL),A
170          POP   BC
171          POP   HL
172          DEC   B
173          DEC   B
174          INC   HL
175          DEC   C
176          JR    NZ,LAK2
177 LAK3
178          LD    A,$5C
179          LD    (HL),A
180          LD    DE,21
181          ADD   HL,DE
182          LD    A,$9A
183          CALL  LINE
184          LD    A,$2F
185          LD    (HL),A
186          RET
187
188 RARIKABE
189          CALL  PARASET
190          LD    A,$2F
191          LD    (HL),A
192
193          PUSH  HL
194          LD    DE,21
195          ADD   HL,DE
196          LD    A,$9A
197          PUSH  BC
198          CALL  LINE
199          LD    A,$5C
200          LD    (HL),A
201          POP   BC
202          POP   HL
203          DEC   C
204          RET   Z
205          INC   B
206          INC   B
207 RAK2
208          LD    DE,21
209          OR    A
210          SBC   HL,DE
211          INC   HL
212          PUSH  HL
213          LD    A,$2F
214          LD    (HL),A
215          ADD   HL,DE
216          LD    A,$20
217          PUSH  BC
218          CALL  LINE
219          LD    A,$5C
220          LD    (HL),A
221          POP   BC
222          POP   HL
223          INC   B
224          INC   B
225          DEC   C
226          JR    NZ,RAK2
227          RET
228
229 PARASET
230          LD    C,(IX+00)
231          LD    B,(IX+01)
232          CALL  ADRSET
233          LD    B,(IX+02)
234          LD    C,(IX+03)
235          RET
236
237 ;TATE LINE  IN B-ﾅｶﾞｻ,HL-ADRESS,A-CHR CODE
238 LINE
239          LD    (HL),A
240          LD    DE,21
241          ADD   HL,DE
242          DJNZ  LINE
243          RET
244
245 ;ｶｿｳｶﾞﾒﾝ ｱﾄﾞﾚｽ ｹｲｻﾝ
246 ADRSET
247          LD    HL,BAFA
248          LD    E,C
249          LD    D,00
250          ADD   HL,DE
251          LD    A,B
252          OR    A
253          RET   Z
254          LD    DE,21
255 ADR2
256          ADD   HL,DE
257          DJNZ  ADR2
258          RET
259
260 ;ｶｿｳｶﾞﾒﾝ ｸﾘｱ
261 BAFACLS
262          LD    HL,BAFA
263          LD    DE,BAFA+1
264          LD    A,$20
265          LD    (HL),A
266          LD    BC,21*21-1
267          LDIR
268          RET
269
270 ;ｶｿｳｶﾞﾒﾝ ﾃﾝｿｳ
271 TENSOU
272          CALL  TEXTON
273          LD    DE,$D000
274          LD    HL,BAFA
275          LD    A,21
276 TENSOU2
277          LD    BC,21
278          LDIR
279          EX    DE,HL
280          LD    BC,40-21
281          ADD   HL,BC
282          EX    DE,HL
283          DEC   A
284          JR    NZ,TENSOU2
285          CALL  TEXTOFF
286          RET
287
288 ;TEXT VRAM BANK CHANGE
289 TEXTON
290          IN    A,($E8)
291          SET   7,A
292          SET   6,A
293          RES   5,A
294          OUT   ($E8),A
295          RET
296 TEXTOFF
297          IN    A,($E8)
298          RES   7,A
299          OUT   ($E8),A
300          RET
301
302 OKUYUKI   DB    00
303 LARIT     DB    00,00,19,04,04,04,11,03
304           DB    07,07,05,02,09,09,01,01
305 LNAIT     DB    00,04,13,04,00,07,07,07
306           DB    06,09,03,03,09,10,01,01
307 RARIT     DB    17,03,13,04,14,06,07,03
308           DB    12,08,03,02,11,09,01,01
309 RNAIT     DB    17,04,13,04,14,07,07,07
310           DB    12,09,03,03,11,10,01,01
311 MIDTABLE  DB    10,01,09,03,07,07,04,13
312 MAZEDAT   DB    01,01,00,01,01
313           DB    00,00,00,01,00
314           DB    00,01,01,00,01
315
316
317
318
```

THE SENTINEL

●リダイレクトってなぁに

リダイレクトとは「画面もキーボードもプリンタもぜ～んぶファイルだ」と強引に定義してしまったことからくる利点のひとつです。

UNIXなどではプログラムは通常,画面という「ファイル」に文字を書き出します。この代わりにディスク上のファイルに文字を書き出すよう，文字の出力先を変更することを「出力をリダイレクトする」といいます。同様に，標準でキーボードから受け取ることになっている文字をディスク上のファイルから受け取ることにしてしまうことを「入力をリダイレクトする」といいます。キーボードから得た文字列をソートして画面に表示するコマンドの入力を，ファイルにリダイレクトすればファイルをソートして眺めることができますし，出力もリダイレクトしてしまえばソート結果をディスク上に残せるようになるわけです。

リダイレクトを使えば，プログラムの出力をファイルに保存しておき，これを別のプログラムの入力に使うことができます。この作業を一気に行うのが「パイプ」です。２つのプログラムの出力と入力の間にパイプラインを設置し，出力される文字を片っ端から入力に放り込むのです。

第85部
SLANG用リダイレクションライブラリDIO.LIB

●リダイレクション用ライブラリ

リダイレクトやパイプを使うことによって，単純な機能のプログラムをいくつも組み合わせ，複雑な処理をさせることが可能になります。住所録を作っておき，特定の文字が含まれている行を探し出すコマンドを使って東京都に住んでいる人を探し出し，結果を名前の順にソートしたファイルを作るなど朝飯前です。

リダイレクトやパイプの考え方の浸透しているUNIXでは，キーボードから文字を受け取り，それを画面に表示するプログラムが山とあり，ユーザーはこれらを随時組み合わせて目的の処理を行うことが当たり前になっています。

われらがS-OS“SWORD”は，ディスクに対する必要最小限の操作しかサポートしていません。いまや常識となった感のある，「ぜ～んぶファイル」という統一的な考え方は採用されていないのです。そこで用意されたのが今回のDIO. LIBです。

このライブラリを使ってSLANG用のプログラムを作れば，リダイレクト機能が自動的にサポートされます。もちろんこのライブラリはSLANGで書かれています。SLANGはSLANGによって強化されていくというわけですね。

```
 */
    ORG     $3000;
    OFFSET  $B000-$3000;

const argcmax=0,arglmax=0;

$include DIO.LIB

var argc;
array byte argv[argcmax][arglmax];

main()
    var KBFAD;
    [
    dioinit(&argc,argv);
    for(KBFAD=memw[$1F76]; getl(KBFAD)!=-1; print(msx$(KBFAD),/))
        if (inkey(0)==$1B)
            exit;
    dioflush();
    ]
$ NANDEMO:<PIPE
 ¥1:¥2 >M:
 ¥3:<M: ¥4
$
```

●S-OSの系譜(5)

“MACE”再掲載の要望が高まるなかで，次期S-OSの構想が練られていきました。当然次期S-OSはフロッピーディスク対応でなければならない，こんなルーチンがほしい，これら読者から集まった次期S-OSへの要望を吟味しながら，次第に仕様が固まっていったわけです。今回は開発者のひとり泉大介さんに当時の様子をうかがいましょう。

「私が受け持ったのは，MZ-80B/2000/2200用のS-OSとDOSモジュールです。牛嶋さんがMZ-80K/C/1200/700/1500，X1/turboを受け持たれ，そして編集の＠さんが全体の調整など一手に引き受けられました」

「まず最初に懸案事項となったのは，DOSモジュールをどのような機能のものにするかです。ディスクドライブを扱えるだけでいいのか，ファイル入出力をサポートした本格的なものにするのか，テープと共存できるかといったことです。私たちがマシン語を使っていて面白いのは，すべての部分に手が届くからです。ファイルへの１文字出力などの機能を用意すると，OSのサービスを利用するといった感じになってしまって面白くない。そこで必要最小限のものに留めることにしました」

「とはいっても，まったくの無法状態というのも将来のバージョンアップに差し支えるだろうという懸念から，ルールは設けさせてもらいました。ファイルアクセスは，まずファイルをオープンし，アクセスしたら最後にクローズすることというルールです。これは，S-OS“MACE”がいかにもBIOSといった方法でテープを扱っていたため，初心者には扱いが難しいと感じられた，という理由もありました。簡単にしようとしたのです」

「次にテープとの共存です。私の受け持ち分は友人との共同作業となったのですが，この友人が２つのフラグを使い分けるといううまい解決法を示してくれました。この話はまた今度にしましょう」

──S-OS“SWORD”の開発秘話はまだまだありそうですね。来月も引き続きその誕生の様子をお届けすることにしましょう。

全機種共通S-OS"SWORD"要

SLANG用リダイレクションライブラリDIO.LIB

SLANG 用のリダイレクションライブラリが完成しました。基本的な機能はαCのDIOライブラリとほぼ同じで，1988年10月号のSLANG用ファイル入出力ライブラリを使って，便利なオペレーション環境を提供するものです。

Nishimura Susumu
西村　進

C言語ライクな記述を意識したSLANGも1988年10月号のファイル入出力ライブラリによってますますCっぽくなってきました。こうなってくると，もっとC言語のまねごとをしたくなるのが人情というもの。と，いうわけでSLANGから I/O リダイレクションを実現するライブラリを作ってみました。

このリダイレクションというやつは，普段はキーボードから入力してディスプレイに出力するという流れを無理矢理ファイルやプリンタなどに切り替えてしまうというものです。MS-DOSやHuman68kではお馴染みの機能でしょう。

使用状況

全機種共通が大前提のS-OSではありますが，悲しいかなこのプログラムを使うにはディスクドライブ（RAMディスク可）が必要です。テープユーザーの方ごめんなさい。でも一応最後まで読んでくださいね。

ディスクユーザーの方は1987年5月号の変身セットでバッチ機能の追加された"SWORD" とファイル入出力関数を拡張されたSLANG を用意してください。そして，E-MATEなどのエディタからDIO.LIBを入力しセーブしておいてください。

さて，このプログラムはあくまでもライブラリですから，これだけではなにもできません。リダイレクションを実現するにはちゃんとメインプログラムを作って，そこからこれらの関数を呼び出さなければなりません。一般的な呼び出し方は，

```
//
//  Program Start
//
    アドレス宣言 ;
  const argcmax=??, arglmax=??;
  # include DIO. LIB
  var argc, argv [argcmax][arglmax];
          ⋮
main( )
          ⋮
  [
  dioinit(& argc, argv);
          ⋮
  dioflush( );
  ]
          ⋮
// End of Program
```

のようになります。

ここで大切なのはmain( )関数の最初でdioinit( )を呼び出し，プログラムを終了するときに必ずdioflush( )を実行することです。特にdioflush( )を忘れると最悪の場合ファイル破壊という憂き目にあいます。特にstop( )関数には注意してください。

このようにしてライブラリを呼び出すとSLANGの入出力関数のうち，

```
inkey(1)
input( )
getl( )
getlin( )
```

の入力，および，print関数の出力先を切り替えることが可能になります（なお，getl( )を実行中にeof(00H)が入力された場合，文字入力バッファの先頭に1BHが書き込まれます)。

サンプルプログラム

それではこれ以降の解説をやりやすくするためにもサンプルプログラムを紹介しておきましょう。NANDEMO. SLとTAB. SL を入力しコンパイルし，ディスク上にそれぞれNANDEMO，TABという実行ファイルを作っておいてください。

まず，

＃ NANDEMO：＜NANDEMO. SL

としてみてください。NANDEMO.SLの内容が表示されましたね。これは標準状態で

リスト1 NANDEMO.SL

```
 1 /*
 2  *  NANDEMO.SL
 3  *
 4  *    これ１本で何でもできてしまう便利なプログラム
 5  *      TYPE:filename = NANDEMO:<filename
 6  *      COPY:f1 to f2 = NANDEMO:<f1 >f2
 7  *      プリンタ出力  = NANDEMO:<filename >L:
 8  *      超簡易エディタ= NANDEMO:>filename
 9  */
10     ORG     $3000;
11     OFFSET  $B000-$3000;
12
13 const argcmax=0,arglmax=0;
14
15 #include DIO.LIB
16
17 var argc;
18 array byte argv[argcmax][arglmax];
19
20 main()
21     var KBFAD;
22     [
23     dioinit(&argc,argv);
24     for(KBFAD=memw[$1F76]; getl(KBFAD)!=-1; print(msx$(KBFAD),/))
25         if (inkey(0)==$1B)
26             exit;
27     dioflush();
28     ]
```

のキーボード入力がファイルからの入力に切り替わったためです。“<”という記号を使うことで入力先が切り替えられるのです。

次に,

　# NANDEMO：<NANDEMO. SL
　　>NANDEMO2. SL

としてみてください。今度はなにも表示されませんでしたね。ここでディレクトリをとってみてください。NANDEMO2. SLというファイルが生成されているでしょう。試しに,

　# NANDEMO：<NANDEMO2. SL

とするとNANDEMO.SLと同じ内容が表示されましたね。つまりこれは，通常ディスプレイへの出力となっているものが“>”によってファイルへの出力に切り替えられたためです。今度は,

　# NANDEMO：<NANDEMO. SL
　　+NANDEMO3. SL

を実行してみてください。画面にNANDEMO. SLの中身が表示され，新たにNANDEMO3. SLというファイルが作成されていますね。このように出力リダイレクト時の指定に“+”を用いると，切り替えたファイル出力の内容を画面に報告するようにできるのです。

デバイスの指定

ところでファイル名にはデバイス名を含めることもできます。たとえば,

　# NANDEMO：<A：ABC >B：DEF

とすると，ドライブAのABCというファイルをドライブBのDEFというファイルに出力します。デバイス名を省略した場合はデフォルトデバイスにアクセスします。また，入出力に同一ドライブの同一ファイル名を指定してもかまいません。

また，このライブラリでは2つのデバイス名，L：とM：を拡張しています。デバイス名L：はプリンタを示すもので，出力先指定のみ可能です。

もうひとつのM：はS-OSの特殊ワークエリアをシングルファイルのRAMディスクとして使えるようにしたものです。S-OSの特殊ワークはラベルテーブルで使われる以外はあまり活躍していないようなので，テンポラリファイルをここに置いて使うことを意識してこういうものを作りました。もちろん，これ以外の用途に使ってもかまいませんが，特殊ワークの内容を破壊してもいいときだけ使ってください。

使う場合は,

　# NANDEMO：<filename >M：

のように指定します。デバイス名の後ろにはファイル名は必要ありません（つけても無視されます）。

引数

DIO. LIBを使うことによってI/Oリダイレクションに加えてコマンドラインからのプログラムに引数を引き渡せるようになります。ただし，ちまたのCとは違って引数の最大数と最大長をあらかじめ指定しておかなければなりません。先ほどのプログラム書式説明のところにあった,

　const argcmax=??, arglmax=??;

という定数宣言はこのためのものです。こ

リスト2　TAB.SL

```
 1 /*
 2  *  TAB.SL
 3  *
 4  *       スペースを圧縮／展開するプログラム
 5  *
 6  *      TAB:<infile >outfile -T(or -t)  :圧縮
 7  *      TAB:<infile >outfile -D(or -d)  :展開
 8  */
 9      ORG     $3000;
10      OFFSET  $B000-$3000;
11
12 const argcmax=0,arglmax=2;
13
14 #include DIO.LIB
15
16 const TAB=9;
17 var argc;
18 array byte argv[argcmax][arglmax];
19
20 main()
21      [
22      dioinit(&argc,argv);
23      if (argc!=1 or argv[0][0]!='-')
24          abort(14);      (*  Bad Data  *)
25      case(argv[0][1])
26          [
27          'T','t':     mktab();
28          'D','d':     detab();
29          others:      abort(14);  (*  Bad Data  *)
30          ]
31      dioflush();
32      ]
33
34 mktab()
35      var c,spct;
36      [
37      spct=0;
38      while((c=getchar())!=EOF)
39          [
40          if (c==$1B)
41              return;
42          if (c!=' ')
43              case(spct)
44                  [
45                  0:  putchar(c);
46                  1:
47                      [
48                      putchar(' ');
49                      putchar(c);
50                      spct=0;
51                      ]
52                  others:
53                      [
54                      putchar(TAB);
55                      putchar(spct);
56                      putchar(c);
57                      spct=0;
58                      ]
59                  ]
60          else
61              spct++;
62          if (inkey(0)==$1B)
63              return;
64          ]
65      ]
66
67 detab()
68      var c,i;
69      [
70      while((c=getchar())!=EOF)
71          [
72          if (c==$1B)
73              return;
74          if (c==TAB)
75              for(c=getchar(); c!=0; c--)
76                  putchar(' ');
77          else
78              putchar(c);
79          if (inkey(0)==$1B)
80              return;
81          ]
82      ]
```

の指定によって引数の最大個は（argcmax+1）個，最大長はarglmaxバイト（ただしエンドコード＝00Hを除く）に制限され，多すぎる指定をした場合などは無視されます。そして，dioinit( )が実行されるとargv[i][j]には（i+1）個目の引数の第j文字目のキャラクタコードが入るというわけです。argcmaxとarglmaxを大きくしたほうがたくさんの引数を使えますが，その分メモリをくってしまいます。それぞれのプログラムで値を変えてください。

問題はどのような書式で引数を渡すかですが，

　# program：<infile >outfile

のような指定を行うプログラムの場合，プログラム名とコロンのあとなら入力ファイルや出力ファイルの前後を問わずどこに置いてもかまいません。守らなければならないのは，それぞれの引数およびリダイレクション指定をスペースで区切るということです。

時としてスペースを含む引数を渡したいときもありますが，こういった場合は引数全体をアポストロフィで囲ってください。その囲んだ文字列中にアポストロフィを入れたい場合はアポストロフィを重ねます。これと同様に，ファイル名にスペースを含んだものを指定したいときにもアポストロフィで囲むようにします。ただし，この場合はファイル名のみを囲むようにします。

さらにリダイレクションの指定は最初に行われたもののみ有効となりますので，

　# XX:ABC <'A:Data 1' 'jun''89' <DEF

とすれば，入力はAドライブのData 1というファイルに切り替わり，“ABC”，“jun '89”，“<DEF”の3つが引数としてXXに渡されます。

この引数を使ったサンプルがTABです。これはソースファイル中のスペースをREDA用のタブコードに圧縮/展開するためのコマンドで，

　# TAB:<TAB.SL >TAB2.SL −T
　# TAB:<TAB2.SL −D

のように使用します。オプションとして指定されている“−T”はスペースをタブに変換するためのもの，“−D”はタブをスペースに戻すためのものです。

パイプ機能？

さて，C言語が使える処理系では往々にしてパイプ機能というものが使えます。これは，

　prog <infile | anotherprog >outfile

としたとき，progの出力をanotherprogの入力とみなして連続実行するという機能です。残念ながら，DIO.LIBではこの機能はサポートされていません（というよりできなかった）。その代わりといってはなんですが，パイプ機能と同じことをするためのバッチファイルを作りました。リスト3をファイル名PIPEでセーブしてください。

　# PIPE:prog <infile anotherprog >outfile

と指定することで先ほどの説明と同じ処理をすることが可能です（当然，progもanotherprogもDIO.LIBを使ったアプリケーション）。

ただし，引数は渡せませんし，ほかのバッチファイルの中で実行するとその後のバッチファイルは使用できません（S-OSではバッチのネストを許していない）。

追加関数

さて，それでは実際にDIO.LIBを使ったアプリケーションの作成法について解説していきましょう。DIO.LIBをインクルードすると，以下の関数が拡張されます。

●定数

EOF：ファイルエンドコード（00H）

●ファイル記述子

stdin　：標準入力（リダイレクト可）
stdout：標準出力（リダイレクト可）
stderr：常にキーボードから入力，画面に出力
stdprn：常にプリンタに出力

●入出力関数

getc(fp)：fpファイル記述子
　fpが示す入力先から1文字読み込んでその値を返す。
putc(c, fp)
　fpが示す出力先にcを出力する。
getchar( )
　標準入力から1文字読み込んでその値を返す。
putchar(c)
　標準出力にcを出力する。

●リダイレクション関係

dioinit(word argc[ ], argv[arglmax])
　コマンド以下を読み込んでリダイレクション実行，引数の設定を行う。
dioflush( )
　リダイレクションを終了する。必要があればエンドコードを出力しファイルをクローズする。

●特殊RAMディスク関係

mopen(md)：md＝読み書きモード

リスト3　PIPE

```
1  ¥1:¥2 >M:
2  ¥3:<M: ¥4
```

　md＝0のとき読み込み用に，1のとき書き込み用にファイルをオープンする。
mgetc( )
　RAMディスクから1文字読み込む。
mputc(c)
　RAMディスクにcを書き込む。
mclose( )
　ファイルをクローズする。

●その他

abort(errno)：errno＝エラー番号
　プログラムを終了する。その際，エラー番号に従ってエラーメッセージを表示する。事前にdioflush( )を実行する必要はない。

コンフィグレーション

DIO.LIBは先頭にある定数を書き換えることによって少々のコンフィグレーションを行うことが可能です。

最初のSOSDCはSLANG用の組み込み関数が使用しないS-OSの出力ルーチン（SLANGは#PRINTしか使っていない）が呼び出された場合もリダイレクションを行うかどうかのフラグで，値が1の場合，#PRINTS，#LTNL，#MSG，#MSX，#MPRINT，#PRTHX，#PRTHLが呼び出された場合でもリダイレクションを実行します（デフォルトは0）。

2つ目はMEMDISKで，特殊ワークエリアを使ったRAMディスクの使用をどのようにするかのフラグです。アクセスする関数だけが必要ならば1，リダイレクションまで行うなら2，まったく使用しないなら0を指定してください（デフォルトは1）。

注意事項

コマンドラインからの指定ひとつでリダイレクションのできるこのライブラリですが，手軽さゆえに危険もつきまといます。くれぐれも大切なファイルを壊してしまうことのないように以下の点に注意してください。

1)　くどいようですが，プログラムを終了する前に必ずdioflush( )を実行してください（abort( )で終了するとき以外）。

2)　安全のため，どんな場合でもなんらかの方法でプログラムを終了できるようにしておいてください（たとえば，ブレイクキ

▶私は今日からある目的のために残業を始めました。それは年末年始のゲーム発売ラッシュに備えて，少しでも多くゲーム購入資金をたくわえておくということです。しかし，私の仕事はSE（システムエンジニア）。ゲーム1本買うのに何Kバイト分のプログラムを組まなければならないかと考えると悲しくなります。　椎橋　茂（24）東京都

ーの入力を見てabort( )するなど)。

3) ディスクアクセス中はもちろんのこと，プログラムの実行が終了しないうちはディスクを抜いたりしないでください。プログラム実行中のリセットも厳禁です。

4) このライブラリを使った場合は読み込み用にfno＝0，書き込み用にfno＝1を使用するので，ファイル入出力関数は使わないようにしてください。また，SOSDC＝1のときも特殊RAMディスクを操作する新関数を使うことは避けてください。

5)このプログラムは呼び出されたとき S-OSの＃LPSWが0（ディスプレイへの出力）であることを想定しています。そうでないとき出力をプリンタに切り替えるとプリンタの出力はおかしくなります。

6) このライブラリではlocate( )を使用してもファイルの書き出し位置は変わりません。

7) プログラムとファイル入出力関数のワークエリア（C800Hからの1Kバイト）が重ならないように注意してください。どうしても重なる場合はワークエリアをずらしてください。

8) 入出力先に同一デバイス上の同一ファイルを指定した場合は，たとえエラーによってプログラムが終了した場合でも前のファイルの内容が消去されますので注意してください。

＊　＊　＊

盛りだくさんの内容とはいえ，少々プログラムが大きくなってしまいました。しかし，NANDEMOやTAB程度のプログラムならちゃんとトランジェントコマンドとして使える大きさになりますから辛抱してください。

ソースプログラムも最初はもう少し美しかったのですが，RAMディスクやコンフィグレーションを追加しているうちに＃コマンドで分断されて非常に見にくいものになってしまいました。使うまいと思っていたgoto文も使ってしまいましたし，ファイル入出力関数のfnoを0，1両方使ってしまったのでDIO.LIBを使うとうっかりファイル入出力関数を使うこともできません。

しかし，なにはともあれ，これでキーボードから入力し画面に出力するという感覚でプログラムを書けばファイル操作が簡単に行えます。これを使ってファイル操作をするプログラムをたくさん作ってみるもよし，K＆Rなどを参考にしてさらにCに迫ってみるもよし，皆さんどうか活用してください。また，テープユーザーの方もバッチ処理が拡張されていればコマンドラインからのパラメータを渡すことだけはできるはずですのでチャレンジしてみてください。

Profile
◇西村さんは京都府にお住まいの19歳，現在大学2年生です。マイコン歴4.5年のMZ-2500ユーザー。9月号で発表された生物進化シミュレーションBUGSの作者でもあります。

リスト4 DIO.LIB

```
  1 /*
  2  *       DIO.LIB
  3  *            S L A N G 用 I/Oリダイレクションライブラリ
  4  *
  5  */
  6
  7 const   SOSDC=0;    /*  S-OSの#PRINT以外の画面表示ルーチンを使うならば 1 */
  8 const   MEMDISK=1;  /*  S-OSの特殊ワークエリアをディスクとして使うなら 1 */
  9                     /*  アクセスする関数のみを必要とするときは 2          */
 10
 11 /*  USAGE
 12 const   argcmax=??,arglmax=??;
 13 ...
 14 #include DIO.LIB
 15 ...
 16 array byte argv[argcmax][arglmax];
 17 var argc;
 18 ...
 19 main()
 20     [
 21     dioinit(&argc,argv);
 22     ...
 23     ...
 24     dioflush();
 25     ]
 26 ...
 27 */
 28
 29 machine @file(1),@svsec(1),@abort(1),
 30         @getl(),@flget(),@print();
 31 #IF (SOSDC==1)
 32 machine @prints(),@ltnl(),@msg(),@msx(),@mprint(),@prthx(),@prthl();
 33 #ENDIF
 34
 35 const   @getnum=1;
 36 #IF (SOSDC==1)
 37 const   @putnum=7;
 38 #ELSE
 39 const   @putnum=0;
 40 #ENDIF
 41
 42 const   stdin=0,stdout=1,stderr=2,stdprn=3,
 43         @con=0,@disk=1,@diskcon=2,@prn=3,@prncon=4,@mem=5,@memcon=6,
 44         EOF=0;
 45
 46 var @charc=0,@inch=0,@outch=0;
 47
 48 #IF (MEMDISK>0)
 49 const @WKSIZE=constw[$1F68];
 50 var @mropf=0,@mwopf=0,@mrp,@mwp,@mwcnt=0;
 51 #ENDIF
 52
 53 array byte @fname0[19],byte @fname1[19],
 54       byte @job[3]= [
 55                     @con,   (*  stdin   *)
 56                     @con,   (*  stdout  *)
 57                     @con,   (*  stderr  *)
 58                     @prn    (*  stdprn  *)
 59                     ];
 60 array
 61     @getex[@getnum] =
 62         [
 63         %$1FD4,     (*  #GETL   *)
 64         %$2022      (*  #FLGET  *)
 65         ],
 66     @getsv[@getnum],
 67     @putex[@putnum] =
 68         [
 69         %$1FF5      (*  #PRINT  *)
 70 #IF (SOSDC==1)
 71                    ,
 72         %$1FF2,     (*  #PRINTS *)
 73         %$1FEF,     (*  #LTNL   *)
 74         %$1FE9,     (*  #MSG    *)
 75         %$1FE6,     (*  #MSX    *)
 76         %$1FE3,     (*  #MPRINT *)
 77         %$1FC2,     (*  #PRTHX  *)
 78         %$1FBF      (*  #PRTHL  *)
 79 #ENDIF
 80         ],
 81     @putsv[@putnum];
 82
 83 dioinit(word argc[],byte argv[][arglmax])
 84     var byte p[],s0,s1;
 85     [
 86     getreg();
 87     @charc=argc[0]=0;
 88     p=^DE;
 89     p=@spcut(p);
 90     while ((s0=p[0])!=0)
 91         [
 92         s1=p[1];
 93         if (s0=='<' and @inch==0 and s1!=0 and s1!=' ')
 94             [
 95             p=@getarg(p+1,@fname0,19);
 96 #IF (MEMDISK==1)
 97             case(memw[@fname0]-':'*256)
 98                 [
 99                 'M','m':
100                     [
101                     mopen(0);
102                     @job[stdin]=@mem;
103                     ]
104                 others:
105                     [
106 #ENDIF
107                     @fopen(0,@fname0,0);
108                     @job[stdin]=@disk;
109 #IF (MEMDISK==1)
110                     ]
111                 ]
112 #ENDIF
113             @getpatch();
114             ]
115         else if ((s0=='>' or s0=='+')
116                     and @outch==0 and s1!=0 and s1!=' ')
117             [
118             p=@getarg(p+1,@fname1,19);
119             case(memw[@fname1]-':'*256)
120                 [
121                 'L','l':
122                     @job[stdout]=@prn+(s0=='+');
123 #IF (MEMDISK==1)
124                 'M','m':
125                     [
126                     mopen(1);
127                     @job[stdout]=@mem+(s0=='+');
128                     ]
129 #ENDIF
130                 others:
131                     [
132                     @fopen(1,@fname1,3);
133                     @job[stdout]=@disk+(s0=='+');
134                     ]
135                 ]
136             @putpatch();
137             ]
138         else if(argc[0]<=argcmax)
139             p=@getarg(p,argv[argc[0]++],arglmax);
140         else
141             p=@skiparg(p);
142         p=@spcut(p);
```

▶さあみんな！　この冬は「ノーライフキング」上映館で会おう！　P.S.でも僕は受験生……。
鴨居　大吾（18）香川県

```
143         ]
144     ]
145 
146 @spcut(byte p[])
147     [
148     while(p[0]==' ')
149         p++;
150     return(p);
151     ]
152 
153 @getarg(byte p[],byte str[],max)
154     var os;
155     [
156     os=0;
157     if (p[0]=='¥'')
158         [
159         p++;
160         while(p[0]!=0)
161             [
162             if (p[0]=='¥'')
163                 [
164                 p++;
165                 if (p[0]!='¥'')
166                     exit;
167                 ]
168             if (os<max)
169                 str[os++]=p[0];
170             p++;
171             ]
172         ]
173     else
174         while(p[0]!=0 and p[0]!=' ')
175             [
176             if (os<max)
177                 str[os++]=p[0];
178             p++;
179             ]
180     str[os]=0;
181     return(p);
182     ]
183 
184 @skiparg(byte p[])
185     [
186     if (p[0]=='¥'')
187         [
188         p++;
189         while(p[0]!=0)
190             [
191             if (p[0]=='¥'')
192                 [
193                 p++;
194                 if (p[0]!='¥'')
195                     exit;
196                 ]
197             p++;
198             ]
199         ]
200     else
201         while(p[0]!=0 and p[0]!=' ')
202             p++;
203     return(p);
204     ]
205 
206 @fopen(fno,fname,mode)
207     [
208     fopen(fno,fname,mode);
209     getreg();
210     if (^CY)
211         abort(^A);
212     ]
213 
214 @fgetc(fno)
215     var c;
216     [
217     c=fgetc(fno);
218     getreg();
219     if (^CY)
220         abort(^A);
221     return(c);
222     ]
223 
224 @fputc(fno,data)
225     [
226     fputc(fno,data);
227     getreg();
228     if (^CY)
229         abort(^A);
230     ]
231 
232 @fclose(fno)    /*  IYレジスタを保存するための二度手間  */
233     [
234     fclose(fno);
235     ]
236 
237 @lpout(c)
238     [
239     code(
240         [c],            (*  LD      HL,c     *)
241         $7D,            (*  LD      A,L      *)
242         $CD,$DC,$1F     (*  CALL    #LPRNT   *)
243         );
244     getreg();
245     if (^CY)
246         abort(^A);
247     ]
248 
249 #IF (MEMDISK>0)
250 mopen(mode)
251     var s;
252     [
253     if (mode)
254         [       (*  wopen   *)
255         if (@mropf)
256             if ((s=sosw[0])>@WKSIZE-4)
257                 abort(9);       (*  Device Full  *)
258             else
259                 @mwp=s+2;
260         else
261             @mwp=2;
262         @mwopf=1;
263         @mwcnt=0;
264         ]
265     else
266         [       (*  ropen   *)
267         if not(@mfexist())
268             abort(8);           (*  File not Found  *)
269         @mrp=2;
270         @mropf=1;
271         if (@mwopf)
272             if ((s=sosw[0])>@WKSIZE-4)
273                 abort(9);       (*  Device Full  *)
274             else
275                 @mwp=s+2;
276         ]
277     ]
278 
279 @mfexist()          /*  特殊ワークエリアの内容がファイルの体裁を */
280     var i,size;     /*  整えていればtrue、そうでなければfalse    */
281     [
282     if ((size=sosw[0])==0 or size>@WKSIZE-2 or sos[size+1]!=0)
283         return(false);
284     for(i=2; i<size+1; i++)
285         if (sos[i]==0)
286             return(false);
287     return(true);
288     ]
289 
290 mgetc()
291     [
292     if (@mropf==0)
293         abort(12);          (* File not Open    *)
294     if (@mrp>@WKSIZE-1)
295         abort(1);           (* Device I/O Error *)
296     return(sos[@mrp++]);
297     ]
298 
299 mputc(c)
300     [
301     if (@mwopf==0)
302         abort(12);          (* File not Open    *)
303     if (@mwp>@WKSIZE-1)
304         abort(9);           (* Device Full      *)
305     @mwcnt++;
306     sos[@mwp++]=c;
307     ]
308 
309 mclose()
310     var i,j;
311     [
312     if (@mwopf)
313         [
314         j=sosw[0]+2;
315         sosw[0]=@mwcnt;
316         if (@mropf)
317             for (i=2; i<2+@mwcnt; i++,j++)
318                 sos[i]=sos[j];
319         ]
320     @mropf=@mwopf=0;
321     ]
322 #ENDIF
323 
324 abort(errno)
325     [
326     @recover();
327     @abort(errno);
328     ]
329 
330 @abort(1)   /*  Thanks to N.Onuki   */
331     [
332     code($7D);  (*  LD  A,L *)
333     stop();
334     ]
335 
336 @getpatch()
337     [
338     @getsv[0]=memw[@getex[0]];  memw[@getex[0]]=&@getl();
339     @getsv[1]=memw[@getex[1]];  memw[@getex[1]]=&@flget();
340     @inch=1;
341     ]
342 
343 @putpatch()
344     [
345     @putsv[0]=memw[@putex[0]];  memw[@putex[0]]=&@print();
346 #IF (SOSDC==1)
347     @putsv[1]=memw[@putex[1]];  memw[@putex[1]]=&@prints();
348     @putsv[2]=memw[@putex[2]];  memw[@putex[2]]=&@ltnl();
349     @putsv[3]=memw[@putex[3]];  memw[@putex[3]]=&@msg();
350     @putsv[4]=memw[@putex[4]];  memw[@putex[4]]=&@msx();
351     @putsv[5]=memw[@putex[5]];  memw[@putex[5]]=&@mprint();
352     @putsv[6]=memw[@putex[6]];  memw[@putex[6]]=&@prthx();
353     @putsv[7]=memw[@putex[7]];  memw[@putex[7]]=&@prthl();
354 #ENDIF
355     @outch=1;
356     ]
357 
358 getc(io)
359     [
360     if (@inch==1)
361         case(@job[io])
362             [
363             @disk:
364                 return(@fgetc(0));
365 #IF (MEMDISK==1)
366             @mem:
367                 return(mgetc());
368 #ENDIF
369             others:
370                 [
371                 code(
372                     [@getsv[1]],    (*  LD      HL,#FLGET   *)
373                     $CD,$81,$1F,    (*  CALL    [HL]        *)
374                     $6F,            (*  LD      L,A         *)
375                     $26,$00         (*  LD      H,0         *)
376                     );
377                 return;
378                 ]
379             ]
380     else
381         return(inkey(1));       (*  #FLGET  *)
382     ]
```


▶日立の「この木なんの木」の歌が変わりました。前回まで2番の歌詞が3番になって，新しい2番が入ってました。と，いうことは，あの歌は4番まであったようです。奥がふかい……。

坊農　誠（18）福井県

```
383
384 getchar()
385     [
386     return(getc(stdin));
387     ]
388
389 putc(c,io)
390     var j;
391     [
392     if (@outch)
393         [
394         case(j=@job[io])
395             [
396 #IF (MEMDISK==1)
397             @con,@diskcon,@prncon,@memcon:
398 #ELSE
399             @con,@diskcon,@prncon:
400 #ENDIF
401                 if (c!=0)
402                     code(
403                     [c],            (*  LD      HL,c        *)
404                     $7D,            (*  LD      A,L         *)
405                     $F5,            (*  PUSH    AF          *)
406                     [@putsv[0]],    (*  LD      HL,#PRINT   *)
407                     $F1,            (*  POP     AF          *)
408                     $CD,$81,$1F     (*  CALL    [HL]        *)
409                     );
410             ]
411         case(j)
412             [
413             @disk,@diskcon:
414                 @fputc(1,c);
415             @prn,@prncon:
416                 if (c!=0)
417                     @lpout(c);
418 #IF (MEMDISK==1)
419             @mem,@memcon:
420                 mputc(c);
421 #ENDIF
422             ]
423         ]
424     else
425         case(@job[io])
426             [
427             @con:
428                 print(str$(c,1));
429             @prn:
430                 [
431                 prmode(2);
432                 print(str$(c,1));
433                 prmode(0);
434                 ]
435             ]
436     if (io==stdout)
437         @charc++;
438     ]
439
440 putchar(c)
441     [
442     putc(c,stdout);
443     ]
444
445 @recover()
446     var i;
447     [
448     if (@inch)
449         [
450         for(i=0; i<@getnum+1; i++)
451             memw[@getex[i]]=@getsv[i];
452         if (@job[stdin]==@disk)
453             @fclose(0);
454         ]
455     if (@outch)
456         [
457         for(i=0; i<@putnum+1; i++)
458             memw[@putex[i]]=@putsv[i];
459         case(@job[stdout])
460             [
461             @disk,@diskcon:
462                 [
463                 @fclose(1);
464                 @chsize(@fname1);
465                 ]
466             ]
467         ]
468 #IF (MEMDISK==1)
469     mclose();
470 #ENDIF
471     ]
472
473 dioflush()
474     [
475     if (@outch)
476         case(@job[stdout])
477             [
478 #IF (MEMDISK==1)
479             @disk,@diskcon,@mem,@memcon:
480 #ELSE
481             @disk,@diskcon:
482 #ENDIF
483                 putchar(0);
484             ]
485     @recover();
486     ]
487
488 @chsize(byte fname[])
489     var err,sectno;
490     var word dirp[];
491     [
492     @file(fname);   (*  file-name をセット  *)
493     dirp=@schfcb(&sectno)+18;   (*  dirp = FCBの(#SIZE) *)
494     dirp[0]=@charc;
495     @svsec(sectno);             (*  書き換えた部分をセクタ丸ごとセーブ  *)
496     ]
497
498 @file(1)                        (*  ファイル名を(#IBFAD)にセット *)
499     [
500     code(                       (*  HL=fnameへのpointer *)
501         $EB,                    (*  EX      DE,HL       *)
502         $3E,$04,                (*  LD      A,'ASCFILE' *)
503         $CD,$A3,$1F             (*  CALL    #FILE       *)
504         );
505     ]
506
507 @schfcb(word sectno[])
508     [
509     code(
510         $CD,$6B,$27,            (*  CALL    FCBSCH      *)
511         $E5,                    (*  PUSH    HL          *)
512         [sectno],               (*  LD      HL,sectno   *)
513         $73,                    (*  LD      (HL),E      *)
514         $23,                    (*  INC     HL          *)
515         $72,                    (*  LD      (HL),D      *)
516         $E1                     (*  POP     HL          *)
517         );
518     ]
519
520 @svsec(1)
521     [
522     code(                       (*  HL=sectno           *)
523         $EB,                    (*  EX      DE,HL       *)
524         $2A,$64,$1F,            (*  LD      HL,(#DTBUF) *)
525         $3E,$01,                (*  LD      A,1         *)
526         $CD,$03,$20             (*  CALL    #DWTSB      *)
527         );
528     getreg();
529     if(^CY)
530         @abort(^A);
531     ]
532
533 /*  Extended S-OS Sub-Routines  */
534
535 @getl()
536     var c,os,byte buf[];
537     [
538     code(
539         $C5,$D5,$E5     (*  PUSH    BC,DE,HL    *)
540         );
541     getreg();
542     buf=^DE;
543     for(os=0; os<255 and (c=getchar())!='¥n'; os++)
544         [
545         case(c)
546             [
547             EOF,$1B:
548                 [
549                 buf[0]=$1B;
550                 goto brkout;
551                 ]
552             others:
553                 buf[os]=c;
554             ]
555         ]
556     buf[os]=0;
557 brkout:
558     code(
559         $E1,$D1,$C1     (*  POP     HL,DE,BC    *)
560         );
561     ]
562
563 @flget()
564     [
565     code(
566         $C5,$D5,$E5     (*  PUSH    BC,DE,HL    *)
567         );
568     getchar();          (*  HL = getchar()      *)
569     code(
570         $7D,            (*  LD      A,L         *)
571         $E1,$D1,$C1     (*  POP     HL,DE,BC    *)
572         );
573     ]
574
575 @print()
576     [
577     code(
578         $F5,$C5,$D5,$E5 (*  PUSH    AF,BC,DE,HL *)
579         );
580     getreg();
581     putchar(^A);
582     code(
583         $E1,$D1,$C1,$F1 (*  POP     HL,DE,BC,AF *)
584         );
585     ]
586
587 #IF (SOSDC==1)
588 @prints()
589     [
590     code(
591         $F5,$C5,$D5,$E5 (*  PUSH    AF,BC,DE,HL *)
592         );
593     putchar(' ');
594     code(
595         $E1,$D1,$C1,$F1 (*  POP     HL,DE,BC,AF *)
596         );
597     ]
598
599 @ltnl()
600     [
601     code(
602         $F5,$C5,$D5,$E5 (*  PUSH    AF,BC,DE,HL *)
603         );
604     putchar('¥n');
605     code(
606         $E1,$D1,$C1,$F1 (*  POP     HL,DE,BC,AF *)
607         );
608     ]
609
610 @msg()
611     [
612     code(
613         $F5,$C5,$D5,$E5 (*  PUSH    AF,BC,DE,HL *)
614         );
615     getreg();
616     print(msg$(^DE));
617     code(
618         $E1,$D1,$C1,$F1 (*  POP     HL,DE,BC,AF *)
619         );
620     ]
621
622 @msx()
```

THE SENTINEL

▶あ～，英和辞典が日本語辞書のようにX68000で辞書登録して扱えるようにならないかな～。だれか作って。
村上　広人（22）静岡県

```
623     [
624     code(
625         $F5,$C5,$D5,$E5 (* PUSH    AF,BC,DE,HL *)
626         );
627     getreg();
628     print(msx$(^DE));
629     code(
630         $E1,$D1,$C1,$F1 (* POP     HL,DE,BC,AF *)
631         );
632     ]
633
634 @mprint()
635     var os,byte buf[];
636     [
637     code(
638         $E3,            (* EX      (SP),HL     *)
639         $C5             (* PUSH    BC          *)
640         );
641     getreg();
642     print(msx$(buf=^HL));
643     while(buf[os++]!=0)
644         ;
645     code(
646         [buf+os],       (* LD      HL,buf+os   *)
647         $C1,            (* POP     BC          *)
648         $E3             (* EX      (SP),HL     *)
649         );
650     ]
651
652 @prthx()
653     [
654     code(
655         $C5,$D5,$E5     (* PUSH    BC,DE,HL    *)
656         );
657     getreg();
658     print(hex2$(^A));
659     code(
660         $E1,$D1,$C1     (* POP     HL,DE,BC    *)
661         );
662     ]
663
664 @prthl()
665     [
666     code(
667         $C5,$D5,$E5     (* PUSH    BC,DE,HL    *)
668         );
669     getreg();
670     print(hex4$(^HL));
671     code(
672         $E1,$D1,$C1     (* POP     HL,DE,BC    *)
673         );
674     ]
675 #ENDIF
676
677 /*  End of DIO.LIB  */
```

全機種共通システムインデックス

■85年 6 月号
序論　共通化の試み
第 1 部　S-OS"MACE"
第 2 部　Lisp-85インタプリタ
第 3 部　チェックサムプログラム
■85年 7 月号
第 4 部　マシン語プログラム開発入門
第 5 部　エディタアセンブラZEDA
第 6 部　デバッグツールZAID
■85年 8 月号
第 7 部　ゲーム開発パッケージBEMS
第 8 部　ソースジェネレータZING
■85年 9 月号
インタラプト　S-OS番外地
第 9 部　マシン語入力ツールMACINTO-S
第10部　Lisp-85入門(1)
■85年10月号
第11部　仮想マシンCAP-X85
連載　Lisp-85入門(2)
■85年11月号
連載　Lisp-85入門(3)
■85年12月号
第12部　Prolog-85発表
■86年 1 月号
第13部　リロケータブルのお話
第14部　FM音源サウンドエディタ
■86年 2 月号
第15部　S-OS"SWORD"
第16部　Prolog-85入門(1)
■86年 3 月号
第17部　magiFORTH発表
連載　Prolog-85入門(2)
■86年 4 月号
第18部　思考ゲームJEWEL
第19部　LIFE GAME
連載　基礎からのmagiFORTH
連載　Prolog-85入門(3)
■86年 5 月号
第20部　スクリーンエディタE-MATE
連載　実戦演習magiFORTH
■86年 6 月号
第21部　Z80TRACER
第22部　magiFORTH TRACER
第23部　ディスクダンプ&エディタ
第24部　"SWORD" 2000 QD
連載　対話で学ぶ magiFORTH
特別付録　PC-8801版S-OS"SWORD"
■86年 7 月号
第25部　FM音源ミュージックシステム
付録　FM音源ボードの製作
連載　計算力アップのmagiFORTH
特別付録　SMC-777版 S-OS"SWORD"
■86年 8 月号
第26部　対局五目並べ
第27部　MZ-2500版 S-OS"SWORD"
■86年 9 月号
第28部　FuzzyBASIC 発表
連載　明日に向かって magiFORTH
■86年10月号
第29部　ちょっと便利な拡張プログラム
第30部　ディスクモニタ DREAM
第31部　FuzzyBASIC 料理法〈1〉
■86年11月号
第32部　パズルゲーム HOTTAN
第33部　MAZE in MAZE
連載　FuzzyBASIC 料理法〈2〉
■86年12月号
第34部　CASL & COMET
連載　FuzzyBASIC 料理法〈3〉
■87年 1 月号
第35部　マシン語入力ツールMACINTO-C
連載　FuzzyBASIC 料理法〈4〉
■87年 2 月号
第36部　アドベンチャーゲーム MARMALADE
第37部　テキアベ作成ツール CONTEX
■87年 3 月号
第38部　魔法使いはアニメがお好き
第39部　アニメーションツール MAGE
付録　"SWORD" 再掲載と MAGIC の標準化
■87年 4 月号
第40部　INVADER GAME
第41部　TANGERINE
■87年 5 月号
第42部　S-OS"SWORD" 変身セット
第43部　MZ-700用 "SWORD" を QD 対応に
■87年 6 月号
インタラプト　コンパイラ物語
第44部　FuzzyBASIC コンパイラ
第45部　エディタアセンブラ ZEDA-3
■87年 7 月号
第46部　STORY MASTER
■87年 8 月号
第47部　パズルゲーム碁石拾い
第48部　漢字出力パッケージ JACKWRITE
特別付録　FM-7/77版 S-OS"SWORD"
■87年 9 月号
第49部　リロケータブル逆アセンブラ Inside-R
特別付録　PC-8001/8801 版 S-OS"SWORD"
■87年10月号
第50部　tiny CORE WARS
第51部　FuzzyBASIC コンパイラの拡張
第52部　X1turbo 版 S-OS"SWORD"
■87年11月号
序論　神話のなかのマイクロコンピュータ
付録　S-OS の仲間たち
第53部　もうひとつの FuzzyBASIC 入門
第54部　ファイルアロケータ&ローダ
インタラプト　S-OS こちら集中治療室
第55部　BACK GAMMON
■87年12月号
第56部　タートルグラフィックパッケージTURTLE
第57部　X1turbo 版 "SWORD" アフターケア
　　　　ラインプリントルーチン
特別付録　PASOPIA7 版 S-OS"SWORD"
■88年 1 月号
第58部　FuzzyBASIC コンパイラ・奥村版
付録　石上版コンパイラ拡張部の修正
■88年 2 月号
第59部　シューティングゲーム ELFES
■88年 3 月号
第60部　構造型コンパイラ言語 SLANG
■88年 4 月号
第61部　デバッギングツール TRADE
第62部　シミュレーションウォーゲーム WALRUS
■88年 5 月号
第63部　シューティングゲーム ELFES II
第64部　地底最大の作戦
■88年 6 月号
第65部　構造化言語 SLANG 入門(1)
第66部　Lisp-85 用 NAMPA シミュレーション
■88年 7 月号
第67部　マルチウィンドウドライバ MW-1
連載　構造化言語 SLANG 入門(2)
■88年 8 月号
第68部　マルチウィンドウエディタ WINER
■88年 9 月号
第69部　超小型エディタ TED-750
第70部　アフターケア WINER の拡張
■88年10月号
第71部　SLANG 用ファイル入出力ライブラリ
第72部　シューティングゲーム MANKAI
■88年11月号
第73部　シューティングゲーム ELFES IV
■88年12月号
第74部　ソースジェネレータ SOURCERY
■89年 1 月号
第75部　パズルゲーム LAST ONE
第76部　ブロックゲーム FLICK
■89年 2 月号
第77部　高速エディタアセンブラ REDA
特別付録　X1版 S-OS"SWORD"〈再掲載〉
■89年 3 月号
第78部　Z80用浮動小数点演算パッケージSOROBAN
■89年 4 月号
第79部　SLANG 用実数演算ライブラリ
■89年 5 月号
第80部　ソースジェネレータ RING
■89年 6 月号
第81部　超小型コンパイラTTC
■89年 7 月号
第82部　TTC用パズルゲーム　TICBAN
■89年 8 月号
第83部　CP/M用ファイルコンバータ
■89年 9 月号
第84部　生物進化シミュレーションBUGS
■89年10月号
第85部　小型インタプリタ言語TTI
■89年11月号
第86部　TTI用パズルゲーム　PUSH BON!

＊以上のアプリケーションは，基本システムである S-OS "MACE" または S-OS"SWORD" がないと動作しませんので ご注意ください。

掲載プログラムを利用するために
Oh!X 標準入力ツール MACINTO-C

Oh!Xのリストページに掲載されているダンプリストはMACINTO-Cというツールで出力されています。掲載プログラムを利用する際にはこのツールを使用されることをおすすめします。

編集室

●ダンプリスト

マシン語プログラムのリストは通常ダンプリストという形で掲載され，Oh!Xでは図1のような形式のダンプリストを採用しています。これはMACINTO-Cというマシン語入力ツールを使用して出力されたものですがOh!Xでは基本的に横8バイト，縦16バイト，CRC付きの形式でダンプリストを掲載します。以下はこのMACINTO-Cを使ったマシン語の入力方法です。その他の入力ツール（各機種のマシン語モニタなど）を使うときも考え方は同じです。

マシン語のプログラムやデータは16進数で番地をつけられたアドレス空間に1バイト（16進2桁）ずつ格納されています。たとえば，図1のダンプリストは32DB$_{H}$番地から335A$_{H}$番地までのリストで32DB$_{H}$番地にC3$_{H}$，32DC$_{H}$にF4$_{H}$……という順に入力していきます。最初のアドレス部分といちばん右の5A$_{H}$というのは入力する必要はありません。

●チェックサム

マシン語プログラムに入力ミスがあるとかなり高い確率で暴走してしまいます。CPUはマシン語実行時にエラーを返すといったことは一切行いません。というのもCPUにとってはプログラムの実行も暴走もたいした違いはないのです。

しかし，プログラムを入力するのは人間ですから，必ず入力ミスをおかしてしまいます。これを検出するのがチェックサムです。ダンプリストのいちばん右端の列（横サム），いちばん下の行（縦サム）がチェックサムを表しています。これらはダンプされたプログラムを数値の集まりとみなして縦横に足し合わせ，その値を16進で表示したときの下2桁の数字となっています。そしていちばん下の右端にある4桁の16進数はCRCチェックバイトと呼ばれるもので，そのブロックのデータを特殊な割り算で計算したときの余りの値を示しています。

もし，ダンプ入力中に1カ所誤りがあったとすると，当然誤った個所の横サムと縦サム，CRCチェックバイトも違った値になることが考えられます。プログラムの入力が終わったら実行させる前にまずCRC，次に縦横のチェックサムを確認してください。これらがすべて合っていれば，入力ミスはまずないと考えてよいでしょう。

●MACINTO-Cの入力

さて実際にマシン語を入力するときに注意すべきこととして，マシン語プログラムの格納されるアドレスの確保があります。特にBASICから入力するときにはCLEAR/LIMITまたはNEWON文を使って，マシン語エリアを確保しなければなりません。例としてマシン語入力ツールMACINTO-CをBASICから入力してみましょう。

MACINTO-Cには3000$_{H}$版とB000$_{H}$版の2種類があります。まず，B000$_{H}$版を入力します。BASICを起動し，

　　NEWON &HB400

または，

　　LIMIT &HB000

を実行しマシン語エリアを確保します。MON/BYEコマンドでマシン語モニタに移りMコマンドなどでリスト2を打ち込みます。詳しくは各機種のマニュアルを参照してください。

すべて打ち込んだらBASICに戻りセーブします。ただし，これはS-OS用のものですので，各機種のBASICなどから使用することはできません。そこで，各機種用サブルーチンのB000$_{H}$版をいま打ち込んだものと重ねて入力します。

ここでリスト15のチェックサムプログラムを使って縦サムと横サムの部分を合わせてください。なお，MACINTO-Cは内部にワークエリアを持っていますので自分自身のチェックサムを取っても正しく表示されません。また3000$_{H}$版はBASICを破壊しないと入力できませんのでディスクしか使用できない人でS-OSなどをお持ちでない人は注意してください。

●使用方法

BASIC上なら，

　　CALL〈先頭アドレス〉

モニタ上なら，

　　G〈先頭アドレス〉

または，

　　J〈先頭アドレス〉

というようにしてMACINTO-Cを起動します。

すると，入力開始アドレスを聞いてきますので各ダンプリストの先頭のアドレスを入力してください。すると指定したアドレスからのダンプリストが表示されます。この状態をダンプモードと呼び，大まかにメモリの状態を見るときに使用します。

ダンプモードでは以下のコマンドが使用できます。

T	1ブロック前を表示
G	1ブロック後ろを表示
S	スタート画面に戻る
P	プリントモードへ
E	エディットモードへ
CLR	ブロックを0で埋める

メモリの内容を書き換えるときはEキーを押してエディットモードに入ってくださ

図1　ダンプリストの形式

```
32DB C3 F4 1F C3 F1 1F C3 EE : 5A
32E3 1F C3 E5 1F C3 D9 1F C3 : 64
32EB D6 1F C3 1A 33 C3 D0 1F : B7
32F3 C3 CD 1F C3 C1 1F C3 BE : D3
32FB 1F C3 B5 1F C3 B2 1F C3 : 0D
3303 18 20 C3 1E 20 C3 11 33 : 40
330B C3 17 33 C3 21 33 3E 0C : 6E
3313 CD F4 1F C9 FE 0C C9 ED : 69
331B 5B 76 1F C3 D3 1F C9 C1 : 2F
3323 C9 C5 47 3E 20 CD 11 33 : 44
332B 78 C1 C9 F5 3A F8 33 B7 : 13
3333 3E 0A C4 7C 33 C5 01 78 : F9
333B 17 DF C1 F1 C9 7C CD 45 : FF
3343 33 7D C5 4F CD 4F 33 CD : E0
334B 4F 33 C1 C9 06 04 CB 11 : F2
3353 8F 10 FB E6 0F C6 30 FE : 83
---------------------------------
SUM: 44 36 E5 E9 B5 CC B5 C1 7318
```

図2　CRCが変わる

```
B200 00 CD F9 B2 CD DE B2 7E : 53
B208 CD F6 B2 7E 83 5F 7E 23 : 76
B210 E3 86 77 23 E3 10 ED E3 : C6
B218 E1 F1 B7 28 0C 3D CD DE : A5
B220 B2 CD DE B2 CD DE B2 18 : 84
B228 F1 CD DE B2 3E 3A CD DB : 6E
B230 B2 CD DE B2 7B CD F6 B2 : FF
B238 C3 E1 B2 C5 01 0F 08 CD : 00
B240 69 B1 C1 18 02 0E 02 61 : 66
B248 2E 05 CD 05 B3 CD ED B2 : 24
B250 CD 02 B3 4C 0D 1A FE 1B : 0E
B258 C8 CD FF B2             : 46
---------------------------------
SUM: D5 07 65 71 88 73 54 02 6FE1
```

```
B200 00 CD F9 B2 CD DE B2 7E : 53
B208 CD F6 B2 7E 83 5F 7E 23 : 76
B210 E3 86 77 23 E3 10 ED E3 : C6
B218 E1 F1 B7 28 0C 3D CD DE : A5
B220 B2 CD DE B2 CD DE B2 18 : 84
B228 F1 CD DE B2 3E 3A CD DB : 6E
B230 B2 CD DE B2 7B CD F6 B2 : FF
B238 C3 E1 B2 C5 01 0F 08 CD : 00
B240 69 B1 C1 18 02 0E 02 61 : 66
B248 2E 05 CD 05 B3 CD ED B2 : 24
B250 CD 02 B3 4C 0D 1A FE 1B : 0E
B258 C8 CD FF B2 00 00 00 00 : 46
B260 00 00 00 00 00 00 00 00 : 00
B268 00 00 00 00 00 00 00 00 : 00
B270 00 00 00 00 00 00 00 00 : 00
B278 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: D5 07 65 71 88 73 54 02 9F90
```

い。先頭のデータ部分にカーソルが点滅しますので，カーソルを移動させて入力/修正が可能です。データはリターンキーで行ごとに登録します。エディット後はブレイクキーでダンプモードに帰ってください。

●プリントモード

ダンプモードでPキーを押すことによりプリントモードに入ります。このモードに入るとSTART ADRS, END ADRS, PRINTER ON(Y/N)と聞いてきますので，順に適当なものを答えていってください。

このモードには2つの使い方があります。まず，ひとつはMACINTO-Cの出力をプリンタに印字すること。もうひとつは1ブロックに満たないブロックのCRCを計算することです。CRCは仕様上の問題から図2のようなことが起こります。このようなときはこのモードを使ってCRCを確認してください。

ダンプ出力中はスペースキーで一時停止，ブレイクで中断します。

●終了

各モードからはブレイクでスタート画面に戻ります。さらにブレイクすることにより，モニタまたはMACINTO-Cを呼び出したシステムに戻ります。どちらに戻るかは機種によって異なります。

●使用上の注意

MACINTO-Cは次のシステム上で動くように作ってあります。

S-OS　S-OS“SWORD”
MZ-80K/C/1200　ROMモニタ
MZ-700/1500　MZ-700用ROMモニタ
MZ-80B/2000　SB-1520,
　　MZ-1Z001M
MZ-2500　BIOS ROM
X1　BASICモニタ
X1turbo　turboBASIC起動後のROMモニタ

また，一般的な注意として入力を途中でやめてセーブしておくとき，以下の機種では実行アドレスを次のようにしてください。

MZ-80K/C/1200/700→0000
MZ-1500→E804
MZ-80B/2000→指定しない

リスト1　MACINTO-C(3000H)

```
3000 CD 08 33 11 89 32 CD E4 : 85
3008 32 CD ED 32 1A FE 1B CA : 1B
3010 0E 33 21 0C 00 19 EB 1A : 8C
3018 FE 50 CA 94 30 FE 70 20 : 6A
3020 05 3E 50 CA 94 30 CD FF : ED
3028 32 38 D5 22 7D 32 CD E1 : BE
3030 32 21 00 00 CD 05 33 11 : 69
3038 96 32 CD E4 32 CD E1 32 : 8B
3040 CD E1 32 01 0F 08 CD 69 : 2E
3048 31 CD F3 32 28 B2 CD F0 : BA
3050 32 FE 53 28 AB FE 54 20 : C8
3058 0E 2A 7D 32 11 80 00 B7 : 2F
3060 ED 52 22 7D 32 18 DC FE : 02
3068 47 20 0C 2A 7D 32 11 80 : DD
3070 00 19 22 7D 32 18 CC CD : 9B
3078 0B 33 20 0F 2A 7D 32 5D : A3
---------------------------------
SUM: 87 B5 62 73 E1 92 CA E3 883F

3080 54 13 01 7F 00 36 00 ED : 0A
3088 B0 18 B8 FE 45 20 05 CD : B5
3090 45 32 18 AF FE 50 20 B1 : 5D
3098 CD 08 33 CD EA 32 11 89 : 8B
30A0 32 CD E4 32 CD ED 32 1A : 1B
30A8 FE 1B CA 00 30 21 0C 00 : 40
30B0 19 EB CD FF 32 38 E4 22 : 40
30B8 7D 32 11 BD 32 CD E4 32 : 92
30C0 CD ED 32 1A FE 1B 28 D3 : 1A
30C8 21 0C 00 19 EB CD FF 32 : 2F
30D0 38 E8 E5 ED 5B 7D 32 B7 : B3
30D8 ED 52 E1 38 BE 22 7F 32 : E9
30E0 11 CA 32 CD E4 32 CD ED : AA
30E8 32 1A FE 1B 28 AD 21 10 : 6B
30F0 00 19 EB 1A E6 DF FE 59 : 3A
30F8 CC E7 32 CD E1 32 2A 7D : 6C
---------------------------------
SUM: FE 81 D5 0E 63 62 2A 23 6DB0

3100 32 11 80 00 19 EB 2A 7F : 70
3108 32 23 B7 ED 52 38 39 F5 : B1
3110 01 0F 08 CD 6F 31 CD E1 : 33
3118 32 2A 7D 32 11 80 00 19 : B5
3120 22 7D 32 F1 CA 9B 30 CD : 24
3128 F3 32 CA 9B 30 CD F0 32 : A9
3130 FE 20 20 CA CD F0 32 47 : 3E
3138 B7 28 F9 CD F3 32 CA 9B : 2F
3140 30 78 FE 20 20 B8 18 EC : A2
3148 2A 7F 32 ED 5B 7D 32 B7 : 89
3150 ED 52 23 7D 0E FF 0C D6 : CE
3158 08 28 02 30 F9 C6 08 47 : 70
3160 CD 6F 31 CD E1 32 C3 9B : AB
3168 30 21 00 02 CD 05 33 C5 : 1D
3170 C5 21 81 32 36 00 11 82 : 62
3178 32 01 07 00 ED B0 2A 7D : 7E
---------------------------------
SUM: A4 87 DF CA F8 3F DB 6E BA4A

3180 32 C1 C5 79 B7 28 08 06 : 1E
3188 08 CD F6 31 0D 20 F8 C1 : E2
3190 CD F6 31 3E 2D 06 21 CD : 53
3198 DB 32 10 FB CD E1 32 11 : 09
31A0 B8 32 CD E4 32 21 81 32 : A1
31A8 06 08 CD DE 32 7E 23 CD : 59
31B0 F6 32 10 F6 CD DE 32 C1 : CC
31B8 79 87 87 87 80 47 2A 7D : 7C
31C0 32 56 5A 23 05 28 27 5E : B7
31C8 23 05 28 22 D5 1E 80 D9 : BE
31D0 E1 D9 7E A3 28 01 37 D9 : 14
31D8 ED 6A 30 08 3E 10 AC 67 : F0
31E0 3E 21 AD 6F D9 CB 0B 30 : 5A
31E8 E9 23 10 E6 D9 EB EB CD : 7E
31F0 F9 32 CD E1 32 C9 3E 08 : 1A
31F8 90 F5 E5 21 81 32 E3 1E : 3F
---------------------------------
SUM: E2 B2 CC 69 14 FB F4 7C 7DB6

3200 00 CD F9 32 CD DE 32 7E : 53
3208 CD F6 32 7E 83 5F 7E 23 : F6
3210 E3 86 77 23 E3 10 ED E3 : C6
3218 E1 F1 B7 28 0C 3D CD DE : A5
3220 32 CD DE 32 CD DE 32 18 : 04
3228 F1 CD DE 32 3E 3A CD DB : EE
3230 32 CD DE 32 7B CD F6 32 : 7F
3238 C3 E1 32 C5 01 0F 08 CD : 80
3240 69 31 C1 18 02 0E 02 61 : E6
3248 2E 05 CD 05 33 CD ED 32 : 24
3250 CD 02 33 4C 0D 1A FE 1B : 8E
3258 C8 CD FF 32 38 DD 13 06 : F4
3260 08 1A FE 20 20 03 13 18 : 8E
3268 F8 CD FC 32 38 CD 77 23 : 92
3270 10 EF 0C C5 01 0F 08 CD : B5
3278 69 31 C1 18 CA 00 00 00 : 3D
---------------------------------
SUM: 4E 8E AC 20 63 2F F9 10 9DC9

3280 00 00 00 00 00 00 00 00 : 00
3288 00 53 54 41 52 54 20 41 : EF
3290 44 52 53 3D 24 00 41 44 : CF
3298 52 53 20 2B 30 20 2B 31 : 9C
32A0 20 2B 32 20 2B 33 20 2B : 46
32A8 34 20 2B 35 20 2B 36 20 : 55
32B0 2B 37 20 3A 53 55 4D 00 : B1
32B8 53 55 4D 3A 00 45 4E 44 : 06
32C0 20 20 20 41 44 52 53 3D : C7
32C8 24 00 50 52 49 4E 54 45 : F6
32D0 52 20 4F 4E 20 28 59 2F : DF
32D8 4E 29 00 C3 F4 1F C3 F1 : 01
32E0 1F C3 EE 1F C3 E5 1F C3 : 79
32E8 D9 1F C3 D6 1F C3 1A 33 : C0
32F0 C3 D0 1F C3 CD 1F C3 C1 : E5
32F8 1F C3 BE 1F C3 B5 1F C3 : 19
---------------------------------
SUM: 26 AD DE ED 57 CF 5B 61 391F

3300 B2 1F C3 18 20 C3 1E 20 : CD
3308 C3 11 33 C3 17 33 C3 21 : F8
3310 33 3E 0C CD F4 1F C9 FE : 24
3318 0C C9 ED 5B 76 1F C3 D3 : 48
3320 1F C9                   : E8
---------------------------------
SUM: D3 00 EF 03 A1 34 6D 12 9358
```

リスト2　MACINTO-C(B000H)

```
B000 CD 08 B3 11 89 B2 CD E4 : 85
B008 B2 CD ED B2 1A FE 1B CA : 1B
B010 0E B3 21 0C 00 19 EB 1A : 0C
B018 FE 50 CA 94 B0 FE 70 20 : EA
B020 05 3E 50 CA 94 B0 CD FF : 6D
B028 B2 38 D5 22 7D B2 CD E1 : BE
B030 B2 21 00 00 CD 05 B3 11 : 69
B038 96 B2 CD E4 B2 CD E1 B2 : 0B
B040 CD E1 B2 01 0F 08 CD 69 : AE
B048 B1 CD F3 B2 28 B2 CD F0 : BA
B050 B2 FE 53 28 AB FE 54 20 : 48
B058 0E 2A 7D B2 11 80 00 B7 : AF
B060 ED 52 22 7D B2 18 DC FE : 82
B068 47 20 0C 2A 7D B2 11 80 : 5D
B070 00 19 22 7D B2 18 CC CD : 1B
B078 0B B3 20 0F 2A 7D B2 5D : A3
---------------------------------
SUM: 07 35 62 F3 E1 92 CA 63 B4AF

B080 54 13 01 7F 00 36 00 ED : 0A
B088 B0 18 B8 FE 45 20 05 CD : B5
B090 45 B2 18 AF FE 50 20 B1 : DD
B098 CD 08 B3 CD EA B2 11 89 : 8B
B0A0 B2 CD E4 B2 CD ED B2 1A : 9B
B0A8 FE 1B CA 00 B0 21 0C 00 : C0
B0B0 19 EB CD FF B2 38 E4 22 : C0
B0B8 7D B2 11 BD B2 CD E4 B2 : 12
B0C0 CD ED B2 1A FE 1B 28 D3 : 9A
B0C8 21 0C 00 19 EB CD FF B2 : AF
B0D0 38 E8 E5 ED 5B 7D B2 B7 : 33
B0D8 ED 52 E1 38 BE 22 7F B2 : 69
B0E0 11 CA B2 CD E4 B2 CD ED : AA
B0E8 B2 1A FE 1B 28 AD 21 10 : EB
B0F0 00 19 EB 1A E6 DF FE 59 : 3A
B0F8 CC E7 B2 CD E1 B2 2A 7D : 6C
---------------------------------
SUM: FE 81 D5 8E E3 E2 2A A3 7F4A

B100 B2 11 80 00 19 EB 2A 7F : F0
B108 B2 23 B7 ED 52 38 39 F5 : 31
B110 01 0F 08 CD 6F B1 CD E1 : B3
B118 B2 2A 7D B2 11 80 00 19 : B5
B120 22 7D B2 F1 CA 9B B0 CD : 24
B128 F3 B2 CA 9B B0 CD F0 B2 : 29
B130 FE 20 20 CA CD F0 B2 47 : BE
B138 B7 28 F9 CD F3 B2 CA 9B : AF
B140 B0 78 FE 20 20 B8 18 EC : 22
B148 2A 7F B2 ED 5B 7D B2 B7 : 89
B150 ED 52 23 7D 0E FF 0C D6 : CE
B158 08 28 02 30 F9 C6 08 47 : 70
B160 CD 6F B1 CD E1 B2 C3 9B : AB
B168 B0 21 00 02 CD 05 B3 C5 : 1D
B170 C5 21 81 B2 36 00 11 82 : E2
B178 B2 01 07 00 ED B0 2A 7D : FE
---------------------------------
SUM: A4 07 5F CA 78 BF DB EE C42D

B180 B2 C1 C5 79 B7 28 08 06 : 9E
B188 08 CD F6 B1 0D 20 F8 C1 : 62
B190 CD F6 B1 3E 2D 06 21 CD : D3
B198 DB B2 10 FB CD E1 B2 11 : 09
B1A0 B8 B2 CD E4 B2 21 81 B2 : 21
B1A8 06 08 CD DE B2 7E 23 CD : D9
B1B0 F6 B2 10 F6 CD DE B2 C1 : CC
B1B8 79 87 87 87 80 47 2A 7D : 7C
B1C0 B2 56 5A 23 05 28 27 5E : 37
B1C8 23 05 28 22 D5 1E 80 D9 : BE
B1D0 E1 D9 7E A3 28 01 37 D9 : 14
B1D8 ED 6A 30 08 3E 10 AC 67 : F0
B1E0 3E 21 AD 6F D9 CB 0B 30 : 5A
B1E8 E9 23 10 E6 D9 EB EB CD : 7E
B1F0 F9 B2 CD E1 B2 C9 3E 08 : 1A
B1F8 90 F5 E5 21 81 B2 E3 1E : BF
---------------------------------
SUM: E2 B2 4C E9 94 7B F4 FC E2F6

B200 00 CD F9 B2 CD DE B2 7E : 53
B208 CD F6 B2 7E 83 5F 7E 23 : 76
B210 E3 86 77 23 E3 10 ED E3 : C6
B218 E1 F1 B7 28 0C 3D CD DE : A5
B220 B2 CD DE B2 CD DE B2 18 : 84
B228 F1 CD DE B2 3E 3A CD DB : 6E
B230 B2 CD DE B2 7B CD F6 B2 : FF
B238 C3 E1 B2 C5 01 0F 08 CD : 00
B240 69 B1 C1 18 02 0E 02 61 : 66
B248 2E 05 CD 05 B3 CD ED B2 : 24
B250 CD 02 B3 4C 0D 1A FE 1B : 0E
B258 C8 CD FF B2 38 DD 13 06 : 74
B260 08 1A FE 20 20 03 13 18 : 8E
B268 F8 CD FC B2 38 CD 77 23 : 12
B270 10 EF 0C C5 01 0F 08 CD : B5
B278 69 B1 C1 18 CA 00 00 00 : BD
```

```
------------------------------------
SUM: 4E 8E 2C 20 E3 2F F9 10 0269

B280 00 00 00 00 00 00 00 00 : 00
B288 00 53 54 41 52 54 20 41 : EF
B290 44 52 53 3D 24 00 41 44 : CF
B298 52 53 20 2B 30 20 2B 31 : 9C
B2A0 20 2B 32 20 2B 33 20 2B : 46
B2A8 34 20 2B 35 20 2B 36 20 : 55
B2B0 2B 37 20 3A 53 55 4D 00 : B1
B2B8 53 55 4D 3A 00 45 4E 44 : 06
B2C0 20 20 20 41 44 52 53 3D : C7
B2C8 24 00 50 52 49 4E 54 45 : F6
B2D0 52 20 4F 4E 20 28 59 2F : DF
B2D8 4E 29 00 C3 F4 1F C3 F1 : 01
B2E0 1F C3 EE 1F C3 E5 1F C3 : 79
B2E8 D9 1F C3 D6 1F C3 1A B3 : 40
B2F0 C3 D0 1F C3 CD 1F C3 C1 : E5
B2F8 1F C3 BE 1F C3 B5 1F C3 : 19
------------------------------------
SUM: 26 AD DE ED 57 CF 5B E1 6D0D

B300 B2 1F C3 18 20 C3 1E 20 : CD
B308 C3 11 B3 C3 17 B3 C3 21 : F8
B310 B3 3E 0C CD F4 1F C9 FE : A4
B318 0C C9 ED 5B 76 1F C3 D3 : 48
B320 1F C9                   : E8
------------------------------------
SUM: 53 00 6F 03 A1 B4 6D 12 B375
```

リスト3　MZ-80K/C用サブルーチン(3000_H)

```
32DB C3 11 33 C3 21 33 C3 2B : 0C
32E3 33 C3 5E 33 C3 67 33 C3 : A7
32EB 6F 33 C3 B2 33 C3 1B 00 : 28
32F3 C3 1E 00 C3 3F 33 C3 3A : 13
32FB 33 C3 1F 04 C3 B9 33 C3 : 8B
3303 C2 33 C3 C6 33 C3 A9 33 : 50
330B C3 AF 33 C3 CA 33 C5 47 : 71
3313 3A CD 33 B7 78 C4 76 33 : D6
331B CD 12 00 78 C1 C9 C5 47 : ED
3323 3E 20 CD 11 33 78 C1 C9 : 71
332B F5 3A CD 33 B7 3E 0D C4 : F5
3333 76 33 CD 06 00 F1 C9 7C : B2
333B CD 3F 33 7D C5 4F CD 49 : E6
3343 33 CD 49 33 C1 C9 06 04 : 10
334B CB 11 8F 10 FB E6 0F C6 : 31
3353 30 FE 3A 38 02 C6 07 CD : 3C
------------------------------------
SUM: 8B 51 48 69 BC 37 30 C8 6C23

335B 11 33 C9 1A 13 B7 C8 CD : 86
3363 11 33 18 F7 F5 3E 01 32 : B9
336B CD 33 F1 C9 F5 AF 32 CD : 5D
3373 33 F1 C9 C5 0E 00 47 CD : D4
337B 92 33 38 10 78 D3 FF 3E : 95
3383 80 D3 FE 0C CD 92 33 38 : 27
338B 03 AF D3 FE 78 C1 C9 F5 : 7A
3393 DB FE E6 0D B9 28 0C CD : 86
339B 1E 00 20 F4 AF 32 CD 33 : 13
33A3 F1 37 C9 F1 B7 C9 3E 16 : B6
33AB CD 12 00 C9 FE 16 C9 11 : 96
33B3 A3 11 CD 03 00 C9 CD 10 : 2A
33BB 04 D8 13 13 13 13 C9 2A : 1B
33C3 71 11 C9 22 71 11 C9 C3 : 7B
33CB 82 00 00                : 82
------------------------------------
SUM: 88 80 1C AC 69 F0 7C 28 673A
```

リスト4　MZ-700/1500用サブルーチン(3000_H)

```
32DB C3 11 33 C3 25 33 C3 2F : 14
32E3 33 C3 66 33 C3 6F 33 C3 : B7
32EB 77 33 C3 CA 33 C3 C2 33 : 22
32F3 C3 BA 33 C3 47 33 C3 42 : F2
32FB 33 C3 1F 04 C3 D5 33 C3 : A7
3303 DE 33 C3 E2 33 C3 B1 33 : 90
330B C3 B7 33 C3 E6 33 C5 47 : 95
3313 3A EB 33 B7 78 C4 7E 33 : FC
331B D3 E3 CD 12 00 D3 E1 78 : C1
3323 C1 C9 C5 47 3E 20 CD 11 : D2
332B 33 78 C1 C9 F5 3A EB 33 : 82
3333 B7 3E 0D C4 7E 33 D3 E3 : 2D
333B CD 06 00 D3 E1 F1 C9 7C : BD
3343 CD 47 33 7D C5 4F CD 51 : F6
334B 33 CD 51 33 C1 C9 06 04 : 18
3353 CB 11 8F 10 FB E6 0F C6 : 31
------------------------------------
SUM: 54 E6 4A 5C C9 76 B9 0D CA39

335B 30 FE 3A 38 02 C6 07 CD : 3C
3363 11 33 C9 1A 13 B7 C8 CD : 86
336B 11 33 18 F7 F5 3E 01 32 : B9
3373 EB 33 F1 C9 F5 AF 32 EB : 99
337B 33 F1 C9 C5 0E 00 47 CD : D4
3383 9A 33 38 10 78 D3 FF 3E : 9D
338B 80 D3 FE 0C CD 9A 33 38 : 2F
3393 03 AF D3 FE 78 C1 C9 F5 : 7A
339B DB FE E6 0D B9 28 0C CD : 86
33A3 BA 33 20 F4 AF 32 EB 33 : 00
33AB F1 37 C9 F1 B7 C9 3E 16 : B6
33B3 CD 11 33 C9 FE 16 C9 D3 : 8A
33BB E3 CD 1E 00 D3 E1 C9 D3 : 1E
33C3 E3 CD 1B 00 D3 E1 C9 D3 : 1B
33CB E3 11 A3 11 CD 03 00 D3 : 4B
33D3 E1 C9 CD 10 04 D8 13 13 : 89
------------------------------------
SUM: 6A 2A 89 CD 5E 6E E7 64 BBBD

33DB 13 13 C9 2A 71 11 C9 22 : 86
33E3 71 11 C9 D3 E3 C3 AD 00 : 71
33EB 00                      : 00
------------------------------------
SUM: 84 24 92 FD 54 D4 76 22 6F3F
```

リスト5　MZ-80B/2000用サブルーチン(3000_H)

```
32DB C3 11 33 C3 21 33 C3 2B : 0C
32E3 33 C3 5E 33 C3 67 33 C3 : A7
32EB 6F 33 C3 B2 33 C3 C0 33 : 00
32F3 C3 62 05 C3 3F 33 C3 3A : 5C
32FB 33 C3 23 06 C3 C7 33 C3 : 9F
3303 D0 33 C3 D4 33 C3 A9 33 : 6C
330B C3 AF 33 C3 D8 33 C5 47 : 7F
3313 3A DB 33 B7 78 C4 76 33 : E4
331B CD C6 08 78 C1 C9 C5 47 : A9
3323 3E 20 CD 11 33 78 C1 C9 : 71
332B F5 3A DB 33 B7 3E 0A C4 : 00
3333 76 33 CD 2E 0A F1 C9 7C : E4
333B CD 3F 33 7D C5 4F CD 49 : E6
3343 33 CD 49 33 C1 C9 06 04 : 10
334B CB 11 8F 10 FB E6 0F C6 : 31
3353 30 FE 3A 38 02 C6 07 CD : 3C
------------------------------------
SUM: 99 57 67 A1 D4 45 D2 FB D511

335B 11 33 C9 1A 13 B7 C8 CD : 86
3363 11 33 18 F7 F5 3E 01 32 : B9
336B DB 33 F1 C9 F5 AF 32 DB : 79
3373 33 F1 C9 C5 0E 00 47 CD : D4
337B 92 33 38 10 78 D3 FF 3E : 95
3383 80 D3 FE 0C CD 92 33 38 : 27
338B 03 AF D3 FE 78 C1 C9 F5 : 7A
3393 DB FE E6 0D B9 28 0C CD : 86
339B 62 05 20 F4 AF 32 DB 33 : 6A
33A3 F1 37 C9 F1 B7 C9 3E 06 : A6
33AB CD C6 08 C9 FE 06 C9 11 : 42
33B3 AB 10 CD A4 06 1A FE 0B : 55
33BB C0 3E 1B 12 C9 AF CD 01 : 71
33C3 09 C3 32 08 CD 14 06 D8 : C5
33CB 13 13 13 13 C9 2A D1 11 : 21
33D3 C9 22 D1 11 C9 C3 B1 00 : 0A
------------------------------------
SUM: 90 85 79 56 13 BD 7E 1E C290

33DB 00                      : 00
------------------------------------
SUM: 00 00 00 00 00 00 00 00 0000
```

リスト6　MZ-2500用サブルーチン(3000_H)

```
32DB C3 11 33 C3 20 33 C3 2A : 0A
32E3 33 C3 5C 33 C3 65 33 C3 : A3
32EB 6D 33 C3 B6 33 C3 BD 33 : FF
32F3 C3 B0 33 C3 3D 33 C3 38 : D4
32FB 33 C3 C5 33 C3 DD 33 C3 : 84
3303 E2 33 C3 E6 33 C3 AB 33 : 92
330B C3 B3 33 C3 EA 33 C5 47 : 95
3313 3A EB 33 B7 78 C4 74 33 : F2
331B DF 03 78 C1 C9 C5 47 3E : 2E
3323 20 CD 11 33 78 C1 C9 F5 : 28
332B 3A EB 33 B7 3E 0A C4 74 : 8F
3333 33 DF 01 F1 C9 7C CD 3D : 53
333B 33 7D C5 4F CD 47 33 CD : D8
3343 47 33 C1 C9 06 04 CB 11 : EA
334B 8F 10 FB E6 0F C6 30 FE : 83
3353 3A 38 02 C6 07 CD 11 33 : 52
------------------------------------
SUM: E7 DD B3 62 DC 0F 6D BB 9597

335B C9 1A 13 B7 C8 CD 11 33 : 86
3363 18 F7 F5 3E 01 32 EB 33 : 93
336B F1 C9 F5 AF 32 EB 33 F1 : 9F
3373 C9 C5 0E 00 47 CD 90 33 : 73
337B 38 10 78 D3 FF 3E 80 D3 : 23
3383 FE 0C CD 90 33 38 03 AF : 84
338B D3 FE 78 C1 C9 F5 DB FE : A1
3393 E6 0D B9 28 10 C5 AF DF : 37
339B 0D C1 FE 03 20 F0 AF 32 : C0
33A3 EB 33 F1 37 C9 F1 B7 C9 : 80
33AB 3E 0C DF 03 C9 DF 0E C9 : AB
33B3 FE 0C C9 DF 0C D0 3E 1B : E7
33BB 12 C9 C5 AF DF 0D C1 C0 : BC
33C3 AF C9 C5 CD D8 33 38 0B : 58
33CB 87 87 87 87 47 CD D8 33 : 3B
33D3 38 01 B0 C1 C9 1A 13 DF : 7F
------------------------------------
SUM: 3E EC D9 D0 D2 9E 62 A5 5E8B

33DB 15 C9 EB DF 14 EB C9 2A : 9A
33E3 E2 05 C9 22 E2 05 C9 C9 : 4B
33EB 00                      : 00
------------------------------------
SUM: F7 CE B4 01 F6 F0 92 F3 90E6
```

リスト7　X1用サブルーチン(3000_H)

```
32DB C3 11 33 C3 21 33 C3 2B : 0C
32E3 33 C3 5E 33 C3 67 33 C3 : A7
32EB 6F 33 C3 A6 33 C3 0C 03 : 10
32F3 C3 4A 00 C3 3F 33 C3 3A : 3F
32FB 33 C3 5E 11 C3 1F 11 C3 : 1B
3303 B1 33 C3 B5 33 C3 9D 33 : 22
330B C3 A3 33 C3 B9 33 C5 47 : 54
3313 3A BA 33 B7 78 C4 76 33 : C3
331B CD 20 14 78 C1 C9 C5 47 : 0F
3323 3E 20 CD 11 33 78 C1 C9 : 71
332B F5 3A BA 33 B7 3E 0A C4 : DF
3333 76 33 CD 46 14 F1 C9 7C : 06
333B CD 3F 33 7D C5 4F CD 49 : E6
3343 33 CD 49 33 C1 C9 06 04 : 10
334B CB 11 8F 10 FB E6 0F C6 : 31
3353 30 FE 3A 38 02 C6 07 CD : 3C
------------------------------------
SUM: 7A 6C 88 99 BF 9D F0 CB DA01

335B 11 33 C9 1A 13 B7 C8 CD : 86
3363 11 33 18 F7 F5 3E 01 32 : B9
336B BA 33 F1 C9 F5 AF 32 BA : 37
3373 33 F1 C9 C5 D5 5F 01 01 : E8
337B 1A ED 78 E6 08 28 0D CD : 6F
3383 F3 32 20 F5 AF 32 BA 33 : 08
338B 7B D1 C1 C9 0D ED 59 0E : 37
3393 03 3E 0E ED 79 3C ED 79 : 57
339B 18 EE 3E 0C CD 20 14 C9 : 1A
33A3 FE 0C C9 11 00 FF CD 03 : B3
33AB 00 D0 3E 1B 12 C9 2A 0E : 3C
33B3 00 C9 22 0E 00 C9 C9 00 : 8B
------------------------------------
SUM: B0 4B 69 76 EE 37 DD 1B BB8B
```

リスト8　X1turbo用サブルーチン(3000_H)

```
32DB C3 11 33 C3 24 33 C3 2E : 12
32E3 33 C3 64 33 C3 6D 33 C3 : B3
32EB 75 33 C3 B3 33 C3 C1 33 : 08
32F3 C3 AC 33 C3 45 33 C3 40 : E0
32FB 33 C3 D2 33 C3 C7 33 C3 : 7B
3303 EF 33 C3 F3 33 C3 A3 33 : A4
330B C3 A9 33 C3 F7 33 C5 47 : 98
3313 3A F8 33 B7 78 C4 7C 33 : 07
331B C5 01 91 17 DF C1 78 C1 : 47
3323 C9 C5 47 3E 20 CD 11 33 : 44
332B 78 C1 C9 F5 3A F8 33 B7 : 13
3333 3E 0A C4 7C 33 C5 01 78 : F9
333B 17 DF C1 F1 C9 7C CD 45 : FF
3343 33 7D C5 4F CD 4F 33 CD : E0
334B 4F 33 C1 C9 06 04 CB 11 : F2
3353 8F 10 FB E6 0F C6 30 FE : 83
------------------------------------
SUM: B9 7A 2F C1 DB F7 49 18 F9E1

335B 3A 38 02 C6 07 CD 11 33 : 52
3363 C9 1A 13 B7 C8 CD 11 33 : 86
336B 18 F7 F5 3E 01 32 F8 33 : A0
3373 F1 C9 F5 AF 32 F8 33 F1 : AC
337B C9 C5 D5 5F 01 01 1A ED : CB
3383 78 E6 08 28 0D CD AC 33 : 47
338B 20 F5 AF 32 F8 33 7B D1 : 6D
3393 C1 C9 0D ED 59 0E 03 3E : 2C
339B 0E ED 79 3C ED 79 18 EE : 1C
33A3 3E 0C CD 11 33 C9 FE 0C : 2E
33AB C9 C5 01 D5 20 DF C1 C9 : ED
33B3 11 00 FF C5 01 E4 1D DF : B6
33BB C1 D0 3E 1B 12 C9 AF 01 : 75
33C3 F0 1F DF C9 CD D2 33 D8 : 61
33CB 67 CD D2 33 D8 6F C9 C5 : 0E
33D3 CD E5 33 38 0B 87 87 87 : BD
------------------------------------
SUM: 39 DA 00 46 64 69 B7 80 1ED2

33DB 87 47 CD E5 33 38 01 B0 : 9C
33E3 C1 C9 C5 1A 13 01 E7 44 : A8
33EB DF C1 3F C9 2A DF FA C9 : 74
33F3 22 DF FA C9 C9 00       : 8D
------------------------------------
SUM: 49 B0 CB 91 39 18 E2 BD 8DDE
```

リスト9　MZ-80K/C用サブルーチン($B000_H$)

```
B2DB C3 11 B3 C3 21 B3 C3 2B : 0C
B2E3 B3 C3 5E B3 C3 67 B3 C3 : 27
B2EB 6F B3 C3 B2 B3 C3 1B 00 : 28
B2F3 C3 1E 00 C3 3F B3 C3 3A : 93
B2FB B3 C3 1F 04 C3 B9 B3 C3 : 8B
B303 C2 B3 C3 C6 B3 C3 A9 B3 : D0
B30B C3 AF B3 C3 CA B3 C5 47 : 71
B313 3A CD B3 B7 78 C4 76 B3 : D6
B31B CD 12 00 78 C1 C9 C5 47 : ED
B323 3E 20 CD 11 B3 78 C1 C9 : F1
B32B F5 3A CD B3 B7 3E 0D C4 : 75
B333 76 B3 CD 06 00 F1 C9 7C : 32
B33B CD 3F B3 7D C5 4F CD 49 : 66
B343 B3 CD 49 B3 C1 C9 06 04 : 10
B34B CB 11 8F 10 FB E6 0F C6 : 31
```

```
B353 30 FE 3A 38 02 C6 07 CD : 3C
---------------------------------
SUM: 0B D1 48 E9 3C B7 30 C8 7AC4

B35B 11 B3 C9 1A 13 B7 C8 CD : 06
B363 11 B3 18 F7 F5 3E 01 32 : 39
B36B CD B3 F1 C9 F5 AF 32 CD : DD
B373 B3 F1 C9 C5 0E 00 47 CD : 54
B37B 92 B3 38 10 78 D3 FF 3E : 15
B383 80 D3 FE 0C CD 92 B3 38 : A7
B38B 03 AF D3 FE 78 C1 C9 F5 : 7A
B393 DB FE E6 0D B9 28 0C CD : 86
B39B 1E 00 20 F4 AF 32 CD B3 : 93
B3A3 F1 37 C9 F1 B7 C9 3E 16 : B6
B3AB CD 12 00 C9 FE 16 C9 11 : 96
B3B3 A3 11 CD 03 00 C9 CD 10 : 2A
B3BB 04 D8 13 13 13 13 C9 2A : 1B
B3C3 71 11 C9 22 71 11 C9 C3 : 7B
B3CB 82 00 00                : 82
---------------------------------
SUM: 08 80 1C AC 69 F0 FC A8 E71E
```

リスト10　MZ-700/1500用サブルーチン(B000H)

```
B2DB C3 11 B3 C3 25 B3 C3 2F : 14
B2E3 B3 C3 66 B3 C3 6F B3 C3 : 37
B2EB 77 B3 C3 CA B3 C3 C2 B3 : A2
B2F3 C3 BA B3 C3 47 B3 C3 42 : F2
B2FB B3 C3 1F 04 C3 D5 B3 C3 : A7
B303 DE B3 C3 E2 B3 C3 B1 B3 : 10
B30B C3 B7 B3 C3 E6 B3 C5 47 : 95
B313 3A EB B3 B7 78 C4 7E B3 : FC
B31B D3 E3 CD 12 00 D3 E1 78 : C1
B323 C1 C9 C5 47 3E 20 CD 11 : D2
B32B B3 78 C1 C9 F5 3A EB B3 : 82
B333 B7 3E 0D C4 7E B3 D3 E3 : AD
B33B CD 06 00 D3 E1 F1 C9 7C : BD
B343 CD 47 B3 7D C5 4F CD 51 : 76
B34B B3 CD 51 B3 C1 C9 06 04 : 18
B353 CB 11 8F 10 FB E6 0F C6 : 31
---------------------------------
SUM: 54 E6 CA 5C C9 76 B9 0D 247F

B35B 30 FE 3A 38 02 C6 07 CD : 3C
B363 11 B3 C9 1A 13 B7 C8 CD : 06
B36B 11 B3 18 F7 F5 3E 01 32 : 39
B373 EB B3 F1 C9 F5 AF 32 EB : 19
B37B B3 F1 C9 C5 0E 00 47 CD : 54
B383 9A B3 38 10 78 D3 FF 3E : 1D
B38B 80 D3 FE 0C CD 9A B3 38 : AF
B393 03 AF D3 FE 78 C1 C9 F5 : 7A
B39B DB FE E6 0D B9 28 0C CD : 86
B3A3 BA B3 20 F4 AF 32 EB B3 : 00
B3AB F1 37 C9 F1 B7 C9 3E 16 : B6
B3B3 CD 11 B3 C9 FE 16 C9 D3 : 0A
B3BB E3 CD 1E 00 D3 E1 C9 D3 : 1E
B3C3 E3 CD 1B 00 D3 E1 C9 D3 : 1B
B3CB E3 11 A3 11 CD 03 00 D3 : 4B
B3D3 E1 C9 CD 10 04 D8 13 13 : 89
---------------------------------
SUM: EA AA 09 CD 5E 6E 67 E4 50C4

B3DB 13 13 C9 2A 71 11 C9 22 : 86
B3E3 71 11 C9 D3 E3 C3 AD 00 : 71
B3EB 00                      : 00
---------------------------------
SUM: 84 24 92 FD 54 D4 76 22 6F3F
```

リスト11　MZ-80B/2000用サブルーチン(B000H)

```
B2DB C3 11 B3 C3 21 B3 C3 2B : 0C
B2E3 B3 C3 5E B3 C3 67 B3 C3 : 27
B2EB 6F B3 C3 B2 B3 C3 C0 B3 : 80
B2F3 C3 62 05 C3 3F B3 C3 3A : DC
B2FB B3 C3 23 06 C3 C7 B3 C3 : 9F
B303 D0 B3 C3 D4 B3 C3 A9 B3 : EC
B30B C3 AF B3 C3 D8 B3 C5 47 : 7F
B313 3A DB B3 B7 78 C4 76 B3 : E4
B31B CD C6 08 78 C1 C9 C5 47 : A9
B323 3E 20 CD 11 B3 78 C1 C9 : F1
B32B F5 3A DB B3 B7 3E 0A C4 : 80
B333 76 B3 CD 2E 0A F1 C9 7C : 64
B33B CD 3F B3 7D C5 4F CD 49 : 66
B343 B3 CD 49 B3 C1 C9 06 04 : 10
B34B CB 11 8F 10 FB E6 0F C6 : 31
B353 30 FE 3A 38 02 C6 07 CD : 3C
---------------------------------
SUM: 19 D7 67 21 54 C5 D2 7B C034

B35B 11 B3 C9 1A 13 B7 C8 CD : 06
B363 11 B3 18 F7 F5 3E 01 32 : 39
B36B DB B3 F1 C9 F5 AF 32 DB : F9
B373 B3 F1 C9 C5 0E 00 47 CD : 54
B37B 92 B3 38 10 78 D3 FF 3E : 15
B383 80 D3 FE 0C CD 92 B3 38 : A7
B38B 03 AF D3 FE 78 C1 C9 F5 : 7A
B393 DB FE E6 0D B9 28 0C CD : 86
B39B 62 05 20 F4 AF 32 DB B3 : EA
B3A3 F1 37 C9 F1 B7 C9 3E 06 : A6
B3AB CD C6 08 C9 FE 06 C9 11 : 42
B3B3 AB 10 CD A4 06 1A FE 0B : 55
B3BB C0 3E 1B 12 C9 AF CD 01 : 71
B3C3 09 C3 32 08 CD 14 06 D8 : C5
B3CB 13 13 13 13 C9 2A D1 11 : 21
B3D3 C9 22 D1 11 C9 C3 B1 00 : 0A
---------------------------------
SUM: 10 85 79 56 13 BD FE 9E 1444

B3DB 00                      : 00
---------------------------------
SUM: 00 00 00 00 00 00 00 00 0000
```

リスト12　MZ-2500用サブルーチン(B000H)

```
B2DB C3 11 B3 C3 20 B3 C3 2A : 0A
B2E3 B3 C3 5C B3 C3 65 B3 C3 : 23
B2EB 6D B3 C3 B6 B3 C3 BD B3 : 7F
B2F3 C3 B0 B3 C3 3D B3 C3 38 : D4
B2FB B3 C3 C5 B3 C3 DD B3 C3 : 04
B303 E2 B3 C3 E6 B3 C3 AB B3 : 12
B30B C3 B3 B3 C3 EA B3 C5 47 : 95
B313 3A EB B3 B7 78 C4 74 B3 : F2
B31B DF 03 78 C1 C9 C5 47 3E : 2E
B323 20 CD 11 B3 78 C1 C9 F5 : A8
B32B 3A EB B3 B7 3E 0A C4 74 : 0F
B333 B3 DF 01 F1 C9 7C CD 3D : D3
B33B B3 7D C5 4F CD 47 B3 CD : D8
B343 47 B3 C1 C9 06 04 CB 11 : 6A
B34B 8F 10 FB E6 0F C6 30 FE : 83
B353 3A 38 02 C6 07 CD 11 B3 : D2
---------------------------------
SUM: E7 5D 33 E2 DC 8F ED BB 908D

B35B C9 1A 13 B7 C8 CD 11 B3 : 06
B363 18 F7 F5 3E 01 32 EB B3 : 13
B36B F1 C9 F5 AF 32 EB B3 F1 : 1F
B373 C9 C5 0E 00 47 CD 90 B3 : F3
B37B 38 10 78 D3 FF 3E 80 D3 : 23
B383 FE 0C CD 90 B3 38 03 AF : 04
B38B D3 FE 78 C1 C9 F5 DB FE : A1
B393 E6 0D B9 28 10 C5 AF DF : 37
B39B 0D C1 FE 03 20 F0 AF 32 : C0
B3A3 EB B3 F1 37 C9 F1 B7 C9 : 00
B3AB 3E 0C DF 03 C9 DF 0E C9 : AB
B3B3 FE 0C C9 DF 0C D0 3E 1B : E7
B3BB 12 C9 C5 AF DF 0D C1 C0 : BC
B3C3 AF C9 C5 CD D8 B3 38 0B : D8
B3CB 87 87 87 87 47 CD D8 B3 : BB
B3D3 38 01 B0 C1 C9 1A 13 DF : 7F
---------------------------------
SUM: 3E 6C D9 D0 52 1E E2 A5 7DD0

B3DB 15 C9 EB DF 14 EB C9 2A : 9A
B3E3 E2 05 C9 22 E2 05 C9 C9 : 4B
B3EB 00                      : 00
---------------------------------
SUM: F7 CE B4 01 F6 F0 92 F3 90E6
```

リスト13　X1用サブルーチン(B000H)

```
B2DB C3 11 B3 C3 21 B3 C3 2B : 0C
B2E3 B3 C3 5E B3 C3 67 B3 C3 : 27
B2EB 6F B3 C3 A6 B3 C3 0C 03 : 10
B2F3 C3 4A 00 C3 3F B3 C3 3A : BF
B2FB B3 C3 5E 11 C3 1F 11 C3 : 9B
B303 B1 B3 C3 B5 B3 C3 9D B3 : A2
B30B C3 A3 B3 C3 B9 B3 C5 47 : 54
B313 3A BA B3 B7 78 C4 76 B3 : C3
B31B CD 20 14 78 C1 C9 C5 47 : 0F
B323 3E 20 CD 11 B3 78 C1 C9 : F1
B32B F5 3A BA B3 B7 3E 0A C4 : 5F
B333 76 B3 CD 46 14 F1 C9 7C : 86
B33B CD 3F B3 7D C5 4F CD 49 : 66
B343 B3 CD 49 B3 C1 C9 06 04 : 10
B34B CB 11 8F 10 FB E6 0F C6 : 31
B353 30 FE 3A 38 02 C6 07 CD : 3C
---------------------------------
SUM: FA EC 88 19 3F 1D 70 CB 4E2D

B35B 11 B3 C9 1A 13 B7 C8 CD : 06
B363 11 B3 18 F7 F5 3E 01 32 : 39
B36B BA B3 F1 C9 F5 AF 32 BA : B7
B373 B3 F1 C9 C5 D5 5F 01 01 : 68
B37B 1A ED 78 E6 08 28 0D CD : 6F
B383 F3 B2 20 F5 AF 32 BA B3 : 08
B38B 7B D1 C1 C9 0D ED 59 0E : 37
B393 03 3E 0E ED 79 3C ED 79 : 57
B39B 18 EE 3E 0C CD 20 14 C9 : 1A
B3A3 FE 0C C9 11 00 FF CD 03 : B3
B3AB 00 D0 3E 1B 12 C9 2A 0E : 3C
B3B3 00 C9 22 0E 00 C9 C9 00 : 8B
---------------------------------
SUM: 30 4B 69 76 EE 37 DD 9B A633
```

リスト14　X1turbo用サブルーチン(B000H)

```
B2DB C3 11 B3 C3 24 B3 C3 2E : 12
B2E3 B3 C3 64 B3 C3 6D B3 C3 : 33
B2EB 75 B3 C3 B3 B3 C3 C1 B3 : 88
B2F3 C3 AC B3 C3 45 B3 C3 40 : E0
B2FB B3 C3 D2 B3 C3 C7 B3 C3 : FB
B303 EF B3 C3 F3 B3 C3 A3 B3 : 24
B30B C3 A9 B3 C3 F7 B3 C5 47 : 98
B313 3A F8 B3 B7 78 C4 7C B3 : 07
B31B C5 01 91 17 DF C1 78 C1 : 47
B323 C9 C5 47 3E 20 CD 11 B3 : C4
B32B 78 C1 C9 F5 3A F8 B3 B7 : 93
B333 3E 0A C4 7C B3 C5 01 78 : 79
B33B 17 DF C1 F1 C9 7C CD 45 : FF
B343 B3 7D C5 4F CD 45 B3 CD : E0
B34B 4F B3 C1 C9 06 04 CB 11 : 72
B353 8F 10 FB E6 0F C6 30 FE : 83
---------------------------------
SUM: 39 FA 2F C1 5B 77 49 18 F663

B35B 3A 38 02 C6 07 CD 11 B3 : D2
B363 C9 1A 13 B7 C8 CD 11 B3 : 06
B36B 18 F7 F5 3E 01 32 F8 B3 : 20
B373 F1 C9 F5 AF 32 F8 B3 F1 : 2C
B37B C9 C5 D5 5F 01 01 1A ED : CB
B383 78 E6 08 28 0D CD AC B3 : C7
B38B 20 F5 AF 32 F8 B3 7B D1 : ED
B393 C1 C9 0D ED 59 0E 03 3E : 2C
B39B 0E ED 79 3C ED 79 18 EE : 1C
B3A3 3E 0C CD 11 B3 C9 FE 0C : AE
B3AB C9 C5 01 D5 20 DF C1 C9 : ED
B3B3 11 00 FF C5 01 E4 1D DF : B6
B3BB C1 D0 3E 1B 12 C9 AF 01 : 75
B3C3 F0 1F DF C9 CD D2 B3 D8 : E1
B3CB 67 CD D2 B3 D8 6F C9 C5 : 8E
B3D3 CD E5 B3 38 0B 87 87 87 : 3D
---------------------------------
SUM: 39 DA 80 C6 E4 E9 B7 80 7EC6

B3DB 87 47 CD E5 B3 38 01 B0 : 1C
B3E3 C1 C9 C5 1A 13 01 E7 44 : A8
B3EB DF C1 3F C9 2A DF FA C9 : 74
B3F3 22 DF FA C9 C9 00       : 8D
---------------------------------
SUM: 49 B0 CB 91 B9 18 E2 BD AF04
```

リスト15　BASIC版チェックサム(HuBASIC)

```
10 REM CHECK SUM
20 CLS
30     DIM VSUM(7)
40     DEF FNA$(X)=RIGHT$(HEX$(X),2)
50     DEF FNB$(X$)=RIGHT$("0"+X$,2)
60 INPUT "PRINT OUT? Y/N";YORN$
70 INPUT "START ADDRESS";SA$
80 IF YORN$="Y" ELSE 190
90     INPUT "END ADDRESS";EA$
100      D$="LPT:"
110      A1=VAL("&H"+LEFT$(SA$,4))
120      A2=VAL("&H"+LEFT$(EA$,4))
130      PRINT"HIT KEY"
140      DM$=INKEY$
150      WHILE A1<A2
160              GOSUB "CHECK"
170      WEND
180      CLOSE
190 'END IF
200 D$="CRT:"
210 ADR=VAL("&H"+LEFT$(SA$,4))
220 PRINT "'T'=>PREVIOUS  'G'=>NEXT"
230 PRINT "ANY KEY START"
240 REPEAT
250      IN$=INKEY$(1)
260      IF IN$="T" THEN ADR=ADR-128
270      IF IN$="G" THEN ADR=ADR+128
280      A1=ADR
290      GOSUB "CHECK"
300 UNTIL IN$="!"
310 END
320 LABEL "CHECK"
330 OPEN "O",#1,D$+"SUM"
340 FOR I=0 TO 15
350      PRINT#1,RIGHT$("000"+HEX$(A1),4);
360      FOR J=0 TO7
370              M1=PEEK(A1+J)
380              HSUM=HSUM+M1
390              VSUM(J)=VSUM(J)+M1
400              DAT$=HEX$(M1)
410              PRINT#1," ";FNB$(DAT$);
420      NEXT
430      H1$=FNA$(HSUM)
440      HSUM=0
450      PRINT#1," :"FNB$(H1$)
460      A1=A1+8
470 NEXT
480 PRINT#1,STRING$(32,"-")
490 PRINT#1,"SUM:";
500 FOR I=0 TO 7
510      V1$=FNA$(VSUM(I))
520      PRINT#1," ";FNB$(V1$);
530      VSUM(I)=0
540 NEXT
550 PRINT#1
560 PRINT#1
570 CLOSE
580 RETURN
```

Oh!X INDEX '89

特集

特別企画

THE SOFTOUCH

S-OS全機種共通企画

連載・シリーズ

機種別活用＆プログラム

紹介レポート

INFORMATION

PENGUIN INFORMATION CORNER

ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

パソコン新製品2機種
PC-8041/AX286D
シャープ

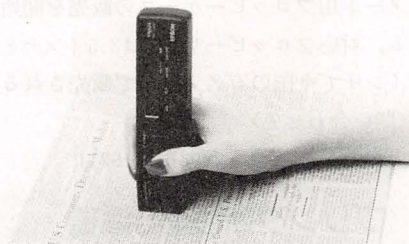
PC-8041

シャープは，業界初の14インチカラー液晶ディスプレイ搭載の32ビットパーソナルワークステーション「PC-8041」と，16ビットAXパソコン「AX286D」の販売を開始する。

PC-8041は，640×480ドットの14インチカラーDST液晶ディスプレイを搭載したIBM PC/AT互換32ビットパソコン。640×480モードで512色中16色，320×200モードで512色中256色が表示できるため，VGAに対応したソフトも使用できる。CPUには80386(20MHz)を採用，オプションで80387も装着することができる。重量は13.5kgでディスプレイとキーボードを一体化できるトランスポータブル型となっている。価格は，40Mバイトハードディスク，3.5インチフロッピーなどの構成で998,000円。

AX286Dは，CPUに80286(12MHz)を採用したAX仕様パソコン。主記憶は標準で640Kバイト，最大13.6Mバイトまで拡張可能。日本語フロントプロセッサVJE-β採用のMS-DOSver3.2を搭載している。価格は，40Mバイトハードディスク内蔵モデルが458,000円。3.5インチフロッピー2基内蔵モデルが278,000円となっている。

〈問い合わせ先〉
シャープ㈱　☎06(621)1221，03(260)1161

OCR感覚のハンディ翻訳機
TRAN PRO-1000
エプソン販売

エプソン販売はハンディ電子翻訳機「サイバートランスレーターTRAN PRO-1000」の販売を開始する。訳したい単語をなぞるのみで和訳でき，従来のようにキーボードから英語を入力する必要がない。また，重量150gで形状がペンタイプになっているため携帯に便利。

文字の認識率は，明朝体で90%，ゴシック体で80%以上。辞書に登録してある単語の数は30889。旅行先などで，英字新聞や雑誌を読んでいて不明な単語があった場合，スピーディーに和訳できる。また，最大52単語まで記憶し，呼び出し復習することも可能。価格は，32,000円で11月下旬発売の予定。

〈問い合わせ先〉
エプソン販売㈱　☎03(377)3500

「ミニ書院」の新製品
WD-A610/A100
シャープ

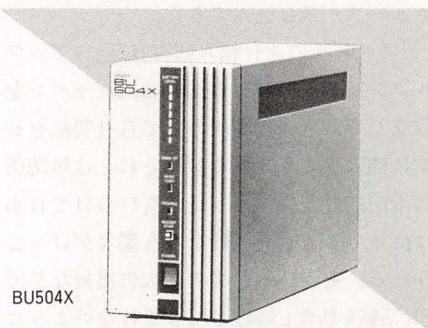
WD-A610

シャープは，ワープロ「ミニ書院」の新シリーズとして，CRT型でAI-V2辞書搭載の「WD-A610」，ローコスト版の「WD-A100」の販売を開始する。

パーソナルワープロの世界では，変換効率の高さや，高品位・高速印字などの基本機能の充実が求められている。同製品は，これらの要求に応えたもので，従来のAI用例辞書に加え，新設計のAI-V2辞書(11万用例)を搭載した。これを，基本辞書(14万語)と併用することによって，変換効率の大幅な向上が図れる。また，48～56ドットから解像度を選択できるプリンタを搭載しており，用途に応じて選択できる。印刷速度は，48ドット時で毎秒50文字となっている。

WD-A100は，おもしろタイピング練習や繰り返し印字機能がついたワープロで学生や初心者をターゲットにしたもの。価格は，WD-A610が145,000円，WD-A100が67,000円となっている。

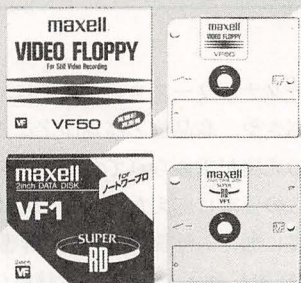
〈問い合わせ先〉
シャープ㈱　☎06(621)1221，03(260)1161

無停電電源装置
BU504X
立石電機

BU504X

立石電機は，パソコンなどで使用できる無停電電源装置「BU504X」の販売を開始する。重量6kgと小型ながら500VAの出力容量を持ち，停電時のトラブルからシステムを保護できる。充電には5時間要し，バックアップ時間は5分間となっている。また，バックアップ事態が生じるとLEDとブザーで警報を出力する。価格は74,800円となっている。

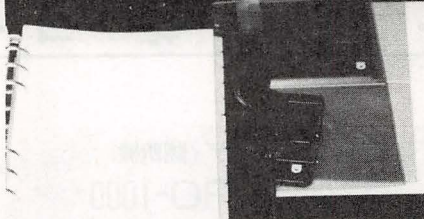
〈問い合わせ先〉
立石電機㈱　☎03(436)7266

マイクロフロッピーディスク VFシリーズ
日立マクセル

VFシリーズ

日立マクセルは，フロッピーディスクの新製品として，2インチ画像記録用の「VF50」と，データ記録用の「VF1」の販売を開始する。VF50は電子スチルカメラの画像記録用で高品質画像を長期にわたって保存できるように設計されている。一方，VF1はノートワープロなどの2インチディスク用で，耐環境性を重視している。価格は，VF50が1枚990円，VF1が1,050円となっている。

〈問い合わせ先〉
日立マクセル㈱　☎03(241)9736

システムノート用フロッピーケース
日本能率協会

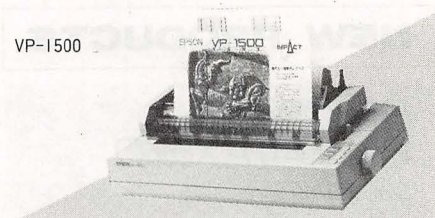
フロッピーケース

日本能率協会は，BindexA5版のシステムノート用フロッピーケースの販売を開始する。対応フロッピーサイズは3.5インチと5インチで全国の有名文具店で販売される。

〈問い合わせ先〉
㈳日本能率協会　☎03(434)6211

プリンタ新製品6機種
エプソン販売

エプソン販売は，10月25日より，ESC/P対応のプリンタ6製品の販売を開始する。販売開始されるのは，カラー印字可能で漢字ゴシックフォントを搭載した132桁ドットインパクトプリンタVP-2050(156,000円)。同じく80カラムタイプのVP-950（126,000円)。インパクト型の普及機VP-1500(136,000円) VP-1350(94,800円)。48ドット熱転写プリンタAP-850(96,800円)。24ドット熱転写のAP-550EX(62,800円)。

〈問い合わせ先〉
エプソン販売㈱　☎03(377)3500

VP-1500

INFORMATION

キャラバンカーでユーザー巡訪
シャープ

この秋からシャープは，先端技術を搭載した2台のキャラバンカーで全国のユーザーを巡訪する。まず，10月16日からOA機器中心の「アクティブOAライナー」を，11月からは，液晶ディスプレイを搭載した「ディスプレイシャトル」をスタートする。こ

Again Watch

今どきの大ニュース

年の瀬に近くなってから，どんどんと大ニュースが飛び込んできた。

「アップルの並行輸入妨害容疑」。アップルコンピュータは日本法人の同ジャパンを設立して総販売代理店として自社製品を日本に輸入販売しているが，それとは無関係に米国内で安売りマックを買いつけて日本に持ってきて販売活動を行う業者がけっこういる。そういった業者の広告掲載などのPR活動を妨害し修理も引き受けないようにした疑いが発覚したため，公正取引委員会が独占禁止法違反の疑いで調査を開始，10月上旬にアップルジャパン，その代理店のキヤノン販売など関係企業の一斉調査に踏み切った。

「ソフト1円受注事件」。富士通が広島市の地図情報システムの受注をたったの1円で落札したことが10月下旬に報道された。こうした問題はいっぱいあり，富士通は長野県でも日本電気との間で同額1円入札合戦を展開したほか，和歌山，埼玉，千葉の自治体，公共機関から1円とか1万円で落札していたことが判明。政府，公正取引委員会も調査に乗り出した。改めてコンピュータ業界のいいかげんさが露呈されるとともに，米国に格好の日本叩き（市場閉鎖の実態）の材料を与えてしまった。

昨年末のAgain Watchで私は「コンピュータのニュースも経済部ネタから社会部ネタに移ってきた。これからはますますそうだろう」と指摘したのだが，まさしくその展開になってきた。

とくに1円受注事件では広島，長野，和歌山……とバレていく様が，昨年のリクルート疑惑議員の発覚報道とオーバーラップするノリさえあった。

これは即ち，コンピュータがそれだけ社会に広く受け入れられてきたことを裏付けている。同時に土地暴騰を機に人々の問題意識が経済フィールドにも広がり，経済大国ニッポンで起こる出来事自体も政治，殺人を題材としたものから金融，産業界を舞台とした経済事件へと移ってきていることも見逃せない。

最近の大きな出来事を列記すると，
- 秀和のスーパー株買い占め
- ソニーがコロンビア映画買収
- 山水電気が英国企業に買収

と本当に産業関係が多い。

おそらく「今年の10大ニュース」の中には経済産業関係の出来事が半分は入るのではなかろうか。もっともトップニュースは「宮崎容疑者による幼女連続殺人事件」，2位は「参議院で野党が過半数議席を獲得」で決まりなのだが。

'89年コンピュータ10大ニュース候補

というようにコンピュータの話題もいよいよメジャーになりつつある昨今，今年のコンピュータ関係の大きな出来事を恒例により，まとめておこう。とはいえあと2カ月も残っているので，エントリーされそうな出来事のタイトルを列記してみる。
- 日電，RISC CPUの独自開発を断念し，米MIPSと提携(2月)。
- インテル，i486を発表。86ファミリーの互換性を2000年まで約束(4月)。

れらは，忙しくてショールームに行けないビジネスマンや主婦が，自宅の近くのイベント広場で気軽に先端技術に触れることを目的としたもの。

OAライナーは，英日翻訳機，音声ワープロ，フルカラー複写機，電子手帳，ファクシミリ，液晶ビジョンなど最新鋭の技術を搭載したキャラバン。展示の際には15坪(約50.0m²)という広さに展開され，ユーザーが実際に使ってみることもできる。ディスプレイシャトルは，液晶，EL，LEDなどの各種フラットパネルディスプレイを展示したキャラバン。こちらは展示してあるのみであるが，「液晶のシャープ」の製品に触れてみる絶好の機会。

〈問い合わせ先〉

シャープ㈱　☎06(621)1221，03(260)1161

アスキーネットサービス拡大
アスキー

アスキーは，同社が運営しているパソコン通信ネットワーク アスキーネットのサービスを拡大する。まず，東京センター直通のアクセスポイントPCSで，電話回線での最高転送速度である9600bpsのモデムを導入する。同時にアスキーカードユーザーへ，9600bps対応のモデムの販売も行う。

また同時に，札幌，仙台，京都，広島，福岡など計13カ所のアクセスポイントを増設する。これにより，地方のアクセス者の通信費が，最大で1/32と大幅な節約が図れる。これら新設アクセスポイントは，300/1200/2400bpsの通信速度をサポートする。

〈問い合わせ先〉

㈱アスキー　☎03(486)9661

BOOK

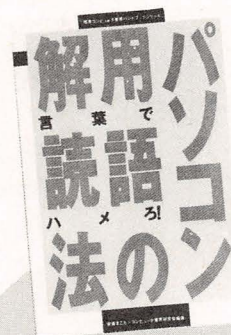
パソコン用語の解説法
ラディク・ラボ

本書は，パソコン用語の正しい解説を行うことを目的に書かれた。「立ち上げる」とはコンピュータを起動することで，RAMは「ラム」と読むなどの基本的な用語について，解説している。新聞や雑誌などでよく登場する用語の解説が多く，コンピュータ用語の辞書を引いても用語が理解できない人は一読。ところで，いたるところで出てくるUNIX関連についての話が比較的面白いので，パソコン以外にUNIXについての話を知りたい人にもお勧め。

コンピュータ業界研究会編著

B6判，202ページ，1,200円

〈問い合わせ先〉

㈱ラディク・ラボ　☎03(235)8061

さよならAgain　1989-12

・ヒューレット・パッカード(HP) がアポロを買収。

・ゲームボーイ，大ヒット(5月)。

・日電，PC-8801打ち止め(？)に向けてPC-98DOを発売（5月)。

・NHKがハイビジョンの定時実験放送を開始(6月)。

・学校用標準パソコンの仕様策定頓座で，TRON計画が後退(6月)。

・東芝が「J-3100ダイナブック」発売。ラップトップが新世代に(6月)。

・米国，スーパーコンなど3品目で日本をスーパー301条制裁候補に(6月)。

・三菱系人工衛星スーパーバードが故障し，衛星通信時代に暗雲(8月)。

・1メガDRAM供給過剰で東芝，日電など各社が増産停止・減産開始(9月)。

・第2KDD 2社，営業開始(10月)。

・OS/2で日本IBM，富士通など11社が連合結成(9月)。

・日電，98に32ビットバスを採用。(10月)

・公取，アップルジャパンの並行輸入妨害容疑を立入り調査(10月)。

・富士通，広島市などでのソフト1円落札事件が相次ぎ発覚(10月)。

と思いついただけでも10個以上ゾロゾロと出てきた。とくに6月と10月に集中していたようで，ちょっと興味深い。

この中のペースメーカーは，依然として日電のようだ。残念ながらシャープが主役に踊り出ることはもはやなさそう。

ワークステーションではいろいろと出来事がありそうだが実はひとつも入らなかった。もう話題の時期から成果を見る時期に入ったと見ていいのか。ただRISC CPU関連でいろいろとトレンドがあった。とくに日電がVシリーズ中心の戦略を断念し，MIPSと提携した件やDECが同じくMIPS 社のRISCチップ搭載マシンを出した件などではRISCの実力が改めて世に知らしめられることになった。

半導体もけっこう話題になった。とくに1メガDRAMの市況を中心とした大手メーカーの値付けと生産状況の推移はなかなか面白い展開を見せた。

通信はハイビジョン放送，第2KDD，衛星放送ISDNなどが話題となった。

'90年代の話題は？

最後に来年以降の見通しについて。

話題の中心はやはり当面は日本電気が握っていると見て間違いない。パソコンと半導体という両戦略商品を押さえているのは強い。同様の理由で東芝も出てきた。

この両社にからむ形で富士通，IBMの展開が興味深い。実際にはこの両社は汎用コンピュータという基盤製品を固めているので実際のポテンシャルは日電とか東芝よりもはるかに上なはずなのだが，影響力がそれより劣っているように見えるのが実にコンピュータビジネスの面白いところである。

ただ繰り返しになるが，社会部ネタはますます増えると見て間違いない。それだけコンピュータが業界ネタから一般社会ネタへと飛躍しているのだ。暗い部屋の隅にコンピュータがひっそりとしていた時代は終わった。いいことなのだろう。

突然だがAgain Watchは今回をもって終了する。ご愛読どうもありがとう。(K.T.)

FILES Oh!X

このインデックスは，タイトル，注記――筆者名，誌名，月号，ページで構成されています。'80年代最後の年も，もう終わりに近づいてきました。忙しい毎日を送ってきた人も，今年1年を振り返ってみてはいかが。

一般

▶図鑑世界のコンピュータちゃん第21回
NCUBE社の並列スーパーコンピュータを例に，並列処理のハイパーキューブなどについて解説している。コンピュータ用語独断的解説大事典も参考になる。――編集部，LOGIN，19号，160-161pp.

▶ネットワーカー・ホリック第8回
ネットからアドベンチャーゲーム誕生，TADLを紹介。パソコン通信版みどりの窓口など。――編集部，LOGIN，19号，234-237pp.

▶図鑑世界のコンピュータちゃん第22回
新日本製鐵の画像処理マシン，FIREPIPを例に画像認識，画像処理について解説。ほかコンピュータ用語独断的解説大事典。――編集部，LOGIN，20号，176-177pp.

▶ネットワーカー・ホリック第9回
読売新聞社が運営をはじめたネット・YOMINETと，アイドルのラジオ番組のネット・パソパソワールド，海外旅行の情報ネット・地球の歩き方ネットを紹介している。――編集部，LOGIN，20号，234-237pp.

▶ハイテク地獄耳
シャープの液晶ディスプレイ付き電話機，子猫物語タムタムを紹介。――編集部，POPCOM，11月号，132p.

▶特集コンピュータグラフィック入門
電脳お絵描き講座のほか，Z'sSTAFF PRO-68K Ver.2.0，サイクロンEXPRESS，Z'sTRIPHONY DIGITAL CRAFTの3本のツールをレポートする。――紀要介/高島早紀，マイコン，11月号，131-157pp.

▶LET'S PROGRAMING！
ソートプログラムの宿題発表。XIturbo，MZ-2000，C言語など。――藤本健，マイコン，11月号，245-254pp.

▶大手予備校/受験生集めの傾向と対策
代々木ゼミナールの受験情報オンラインサービスをレポート。――編集部，マイコン，11月号，279-282pp.

▶ビジネスマンの情報管理術
電子手帳用3.5インチフロッピーディスクCE-70Fの使用法を解説。――塚田洋一，マイコン，11月号，304-307pp.

MZ-80K/C/1200/700/1500

MZ-700/1500（S-BASIC）

▶だぶる　すくろ～る・かにさん どこいくの？・じどうしゃ　げーむ
3本のミニゲームを1本にまとめた。――みすたーえっくす，マイコンBASIC Magazine，11月号，127p.

MZ-700/1500（HuBASIC）

▶功Hu（くんヒュー）
コンピュータと対戦する格闘ゲーム。――松谷泰史，マイコンBASIC Magazine，11月号，128-129pp.

MZ-1500

▶誌上公開質問状
MZ-1500のクイックディスクのヘッド交換について解説している。――編集部，PEGASUSS，マイコンBASIC Magazine，11月号，65-66pp.

▶PA-PA-PA
パックマンタイプのアクションゲーム。移植版。――舟生日出男，マイコンBASIC Magazine，11月号，130-132pp.

MZ-80B/2000/2500/2800

MZ-80B/2500

▶SEA TRAVELER
巨大な物体を倒しに旅に出る。舟を操って敵を倒す，縦スクロールゲーム。――塩浜達也，マイコンBASIC Magazine，11月号，133-135pp.

MZ-2500（BASIC-M25）

▶PUSH and PULL II
魔物に捕まらないように鍵を拾って扉へ。パズルゲーム。――宿久美樹哉，マイコンBASIC Magazine，11月号，136-138pp.

X1/turbo/Z

X1シリーズ

▶誌上公開質問状
Z'sSTAFF-Zで描いたグラフィックをCZ-8PC4でハードコピーするには？　turboZ'sSTAFFの400ラインで描いた絵をturboのBASICで操作するには？――編集部，多田太郎，マイコンBASIC Magazine，11月号，66p.

▶サンプラザ中野さんとスイカ割り
なぜかサンプラザ中野を操り，スイカ割りをするワンキーゲーム。――X1 lettuce，マイコンBASIC Magazine，11月号，164-165pp.

▶DRAGON' X'SPIRIT
ナマズにさらわれたリリーシャ王女を救え！　ドラスピもどきゲーム。――ちろちん，マイコンBASIC Magazine，11月号，166-168pp.

▶3D地図作成プログラム
地図の標高データから3次元のグラフィックを描きおこすプログラム。海面が上がった場合のシミュレートもできる。――沢崎正光，I/O，11月号，117-121pp.

X1＋FM音源ボード（要NEW FM音源ドライバ）

▶サンダークロス6面「Fire Cavern」
ゲームミュージックプログラム。――上田順一，マイコンBASIC Magazine，11月号，200-201pp.

参考文献
I/O　工学社
ASCII　アスキー
テクノポリス　徳間書店
POPCOM　小学館
マイコン　電波新聞社
マイコンBASIC Magazine　電波新聞社
LOGIN　アスキー

コンピュータの本かと思って手に取ったら，そうではなく，ハイパーメディアの本であった。たとえばMacintoshのハイパーカードを思い出す。ハイパーテキストという言葉も有名だ。コンピュータの終焉というタイトルは次の言葉に集約されている。「今やパーソナルコンピュータは，コンピュータの汎用機能を放棄しつつある。……パーソナルコンピュータに残された意味，つまりメディアとして発展させていかなければ，本当に個人のものにはなりえない」汎用性神話がパソコンの呪縛であり，そこから逃れる術はメディアとなる道しかないと著者はいう。

そして，二葉亭四迷からマクルーハン，アラン・ケイ，ウィリアム・ギブソンなどの考えを引きながら，人にとってのメディアは，という問題を論じ始める。ハイパーメディアは人にとってどのようなものとなりうるか。それを人は使いこなせるのか。ユーザーインタフェイス云々を叫びながら，依然として「人間に機械に近づく努力を強いる」今のパソコン環境を考えるとこういった議論はもっとなされるべきだと思う。（K）

コンピューターの終焉　ハイパーメディア・ギャラクシーII　浜野保樹著　福武書店刊
☎03(230)2131　A5判　267ページ　1,600円

書籍の価格は消費税込みです

X1turboシリーズ
▶NEW SOFT
麻雀狂時代SPECIAL II冒険篇を紹介。――編集部，LOGIN，19号，29p.
▶最新ゲーム徹底解剖!!
ウルティマIIを紹介，解説している。――編集部，LOGIN，20号，220-221pp.
▶GAMING WORLD
新着ゲーム，ウルティマIIを紹介。――編集部，テクノポリス，11月号，24p.
▶SOFT RADAR
麻雀狂時代SPECIAL II冒険篇を紹介。――編集部，POPCOM，11月号，29p.
X1turboZシリーズ
▶ALIX
コンピュータとのモビルスーツバトルゲーム。――浪越孝宏，マイコンBASIC Magazine，11月号，169-170pp.

X68000

▶NEW SOFT
ゴルフシミュレーションゲーム，ジャック・ニクラウス・チャンピオンシップ・ゴルフを紹介。――編集部，LOGIN，19号，31p.
▶X68000新聞
3Dソフトの大特集。メタルサイト，サンダーブレード，スーパーハングオン，フラッピー2，ジャック・ニクラウス・チャンピオンシップ・ゴルフ。PDSはX68000用通信ターミナルソフト「Telecom Miki」を紹介。TITLE. SYSの改造など。――編集部，LOGIN，19号，144-149pp.
▶最新ゲーム徹底解剖!!
カードパズルゲーム，琉球を紹介。――編集部，LOGIN，19号，210-211pp.
▶NEW SOFT
スーパー大戦略マップコレクションを紹介。――編集部，LOGIN，20号，19p.
▶X68000新聞
Stationery PRO-68K，夢幻戦士ヴァリスII，Shufflepuck CAFE，リングマスターI・フィリアスノギスの暗雲，メタルサイト。そのほかドット絵の描き方講座，PDS，お料理教室Xなど。――編集部，LOGIN，20号，160-165pp.
▶先取りおすすめゲーム
速くてリアルな3Dゴルフシミュレーションゲーム，遙かなるオーガスタを紹介。その3DシステムのPOLYSYSについて解説もしている。そのほかアークスIIなど。――編集部/加藤英治，テクノポリス，11月号，10-13pp.
▶GAMING WORLD
新着ゲーム，ジャック・ニクラウス・チャンピオンシップ・ゴルフ，リングマスターI・フィリアスノギスの暗雲，シュヴァルツシルト，アルフェイム，ハイブリッドフレーマー・ナイトアームズを紹介。――編集部，テクノポリス，11月号，21-31pp.
▶攻略おすすめゲーム
シミュレーションゲーム，斬[ZAN]陽炎の時代を徹底攻略。各大名のデータリストを掲載。――編集部，テクノポリス，11月号，49-51pp.
▶ゲームがオレを呼んでいる！
斬[ZAN]陽炎の時代を紹介。――稲生達朗，POPCOM，11月号，70-71pp.
▶WE ARE THE X68000 WORLD
新着ゲーム，38万キロの虚空，メタルサイト，ダブルイーグル，フラッピー2と，グラフィックツールのPRISM68K，常駐型ツール・Stationery PRO68Kなどを紹介。――編集部，POPCOM，11月号，84-88pp.
▶X68000の世界へようこそ
ハイエンド・ユーザーが語る愛機の魅力。大阪でX68000ユーザーのBBSを開いている筆者が，その魅力をさまざまな面から紹介。――Cock robin Network大阪……と・とら，マイコンBASIC Magazine，11月号，43-45pp.
▶誌上公開質問状
X-BASICでPEEK/POKE命令に相当するのは？　またRS-232Cを制御する方法は？　内蔵フロッピーディスクで2DDを使えるか？　などの質問に答える。――編集部/多田太郎，マイコンBASIC Magazine，11月号，64-65pp.
▶X68000周辺機器&ソフトカタログ
X68000の周辺機器，実用ソフトを紹介。――編集部，マイコンBASIC Magazine，11月号，カラー折り込み
▶room tennis
1人or2人用テニスゲーム。要JOY STICK。――高橋秀之，マイコンBASIC Magazine，11月号，171-173pp.
▶SPACE BATTLE
モトスタイプの2人用バトルゲーム。宇宙船を操って，相手の船を画面外に追い出せば勝ち。――永井崇博，マイコンBASIC Magazine，11月号，174-176pp.
▶SNATCHERオープニング・テーマ
コナミのゲームミュージックプログラム。――川野俊充，マイコンBASIC Magazine，11月号，186-189pp.
▶チャレンジ！　X68000
新着ゲーム，ジャック・ニクラウス・チャンピオンシップ・ゴルフ，スーパーハングオンとナムコ・オールドゲーム第2弾「モトス」を紹介。――佐久間亮介，マイコンBASIC Magazine，278-279pp.
▶X68000マシン語入門
オリジナルのDIRコマンドを作る。フリーエリア計算機能をカットしてハードディスクでも軽快な反応速度を確保。――高橋雄一，マイコン，11月号，234-239pp.
▶なんでもQ＆A
Human68kV2.0でインタラプトが効かないのはなぜ？　BOOTをRAMに設定したらキーボードの入力ができなくなった，などの質問に答える。――編集部，マイコン，11月号，414-415pp.
▶3Dぴんぽん
マウス操作のラケットでボールを打ち返せ！　PC-88VA版からの移植。――40本場，I/O，11月号，127-129pp.
▶HSL
高速ライン描画サブルーチン。横ラインなら毎秒2000本の描画が可能。――WIZARD N氏，I/O，11月号，130-147pp.
▶Fremes2
I/O6月号の“3Dワイヤーフレームグラフィック”のプログラムを前述HSLを用いて高速化を施したもの。――WIZARD N氏，I/O，11月号，148-154pp.
▶ラジオ・ファクシミリ
X68000に短波ラジオの音声信号を入力して気象放送やファクシミリ通信などをCRTに出力するインタフェイスとプログラム。――Den Den Maru，I/O，11月号，193-198pp.
▶AV WORKSHOP
StationeryPRO-68KとPRISM68Kの2本を紹介。――宮本親一郎，ASCII，11月号，337-341pp.
▶FSV/FLD
65536色の画像を色と輝度に分けて圧縮するユーティリティ。特にデジタイズ画像に威力を発揮する。――後藤英昭，ASCII，11月号，371-376pp.

ポケコン

PC-E200
▶リアルタイムクロックLSIを使う
クロック回路を製作し，PC-E200で制御している。――石川至知，マイコン，11月号，367-370pp.
PC-E500
▶SPACE FIGHTER E500
あなたはいまスペースファイターのコクピットにいる。照準をあわせて敵を撃つ！――石川雅之，マイコンBASIC Magazine，11月号，179p.
PC-1245
▶誌上公開質問状
PC-1245と同じBASICが使えるポケコンは？　プログラムの保存方法は？――編集部，Walking Pockecom，マイコンBASIC Magazine，11月号，66p.
PC-1600K
▶ポケットコンピュータ活用術。
ポケコンの電子手帳化第6回。今回はカレンダー機能プログラムの追加。――塚田洋一，マイコン，11月号，318-321pp.

アメリカのミリテク戦略
本書はコンピュータ科学における各分野の第一人者達のケーススタディを通して，ミリテク（軍事技術）開発の現状やその課題をレポートしたものである。また，コンピュータ技術が兵器システムのもっとも重要な要素でありながら，逆にコンピュータエラーによって生じる偶発核戦争の危険性や，国際的な政治・経済に対する影響も検討している。
デビッド・ベリン/ゲリー・チャプマン編　増田祐司訳　HBJ出版　☎03(234)3911　B5判　350ページ　2,500円

僕らのパソコン10年史
パソコンが世に出始めてから，はや10年がたとうとしている。思えば，この10年間でパソコンは驚くほどの成長を遂げたものだ。本書では，そんな駆け足で過ぎ去ってしまったこの10年間のパソコンの歴史と，前身となったマイクロプロセッサの時代を振り返り，1年ごとにまとめたものである。その年の流行や事件を交えつつ，読みやすく紹介している。また，10編のちょっとしたエッセイも収録されている。
SE編集部編　翔泳社　☎03(263)0447　A5判　250ページ　1,200円

書籍の価格は消費税込みです

QUESTION and ANSWER

文化祭でX68000のビジュアルシェルからダブルクリックで表示するデモを展示しました。が，表示後，グラフィックローダの起動メッセージ（もともとコマンドモードから使うプログラムでした）と“vs.x:～”の文字が残ってみっともないのです。結局，ディスクのメッセージ部分を書き換えましたが，もっといい方法はないものでしょうか。

鳥取県　芹沢　義道

なんとかアイコンメンテナンスでやろうとしてみましたが，どうもいけません。以前，アイコンメンテナンスでファイル名を無効にする際，パラメータで“<nul”を指定することができましたから，今回は“>nul”がききそうに思えたのですがメッセージのリダイレクトはやってくれないようです。

ファイルの起動メッセージを消すだけならvs.xの起動をコマンドシェルから，

A>vs >nul

とすることで対処できます。しかし，この方法でも“vs. x : press mouse button or key” のメッセージは消せません。

ということで，方法として考えられることは次の2つです。

1)　起動ファイル名の前に“/”をつけることで確認なしのプログラム終了モードにして，実行プログラム側でマウスを読んで終了するようにする。

これでは，プログラムの変更が必要ですので，

2)　ビジュアルシェルだけで使うことだけを考え，メモリスイッチのテキストパレットP3をRGBとも0にする。

白い文字を透明色にするわけですね。で，ビジュアルシェルまで見えなくなるのではと心配する方，大丈夫です。vs.xはそのパレットを使っていません。でも使用後の再設定は忘れずに。　　　　（中野　修一）

X1turbo(CZ-852C)ユーザーです。外部メモリ(CZ-8BE2)を購入し，遊びの世界を少し広げようとしております。turboCP/M V2.2(CZ-130SF)のブートドライブを外部メモリにできるとスピードアップが図れて快適になると思うのですが，この方法を教えてください。

愛知県　持田　繁

8ビット用のDOSとして世界最大のユーザー数を誇ったCP/Mに関する質問です。もはや共通メディアとしての価値はあまり高くありませんが，X1シリーズやMZ-2500では言語学習用として安価に提供されていますね。

さて，X1シリーズでCP/Mを使用してソフトウェアの開発をしようと考えるなら，外部メモリ(EMM)はもはや必需品となります。というのもCP/MはOS自体が古いせいもあってか，1レコードを128バイトとして扱うためにディスクの入出力が非常（異常？）に遅いのです。アセンブラやコンパイラを使っていると，ソースリストの入力やオブジェクトコードの出力に要するディスクアクセス時間に膨大な時間をとられてしまいます。その結果アセンブル，コンパイル速度が非常に遅くなってしまい，文法上の些細なミスで発生したアセンブルエラーや，プログラムを開発してデバッグを行うといった一連の単純作業を繰り返すなかで，ディスクランプがついたり消えたりしている時間が長く，とても能率のいい開発環境とはいえません。

ところがEMMを持っていると事態は変わってきます。システムをすべてEMMに転送し，EMM上ですべての入出力を行えばハードディスクのような高速なデータのやり取りが可能となるわけです。

こうなってくると，^Cによってリブートするたびにディスクアクセスを行うのがうっとうしく思えてきます。誰でも一度はEMMをブートドライブに設定できる方法はないかと考えたことがあると思います。

ブートドライブをEMMに設定する方法は本誌1988年6月号の34ページに江川氏が紹介していますが，現在バックナンバーの購入が難しいのではないかと思われますので，ここで改めてその方法を紹介しましょう。これは通常のシステムディスクをブートドライブがEMMであるシステムに書き換えるようにしたものです。X1でCP/Mを利用している方も多いと思われますので，せっかくですからついでにX1用のものも一緒に紹介します。

まず，turboCP/Mを使っている場合，Z80用のアセンブラを使って，

```
LD      A, 6
LD      (ODB1FH), A
XOR     A
LD      (ODB28H), A
RET
```

というプログラムをアセンブルしてください。また，X1CP/Mの場合には，

```
LD      A, 4
LD      (OEA33H), A
RET
```

とします。どちらともファイルネームはEMM.COMとするのを忘れないでください。次に先行キーの設定を行います。EMMをAドライブとしたいCP/Mシステムの入ったシステムをドライブAに入れて，リセットしコマンドモードで，

STARTUP　*

と入力して先行キー設定モードを選択してください。turboCP/Mを使っている場合は以下のように設定します。

COPY ⏎
⏎
AEYNEMM ⏎

また，X1CP/Mの場合には，

COPY ⏎
A ⏎
⏎
E ⏎
⏎
^C　EMM ⏎
^C ⏎

としてください。turboCP/Mでは本来のド

リスト1

```
0000                1 ;         sample pro-X1 Ver B
0000                2
5000                3           ORG      5000H
5000                4
5000 3E 41          5           LD       A,"A"
5002 CD 00 80       6           CALL     BIOS
5005 91 17          7           DW       1791H    ; 1CHR PRINT
5007 C9             8           RET
5008                9
5008               10 ;----------------------------
5008               11
8000               12           ORG      8000H
8000               13           ;
8000 D9            14 BIOS:     EXX
8001 E3            15           EX       (SP),HL
8002 5E            16           LD       E,(HL)
8003 23            17           INC      HL
8004 56            18           LD       D,(HL)
8005 23            19           INC      HL
8006 ED 53 12      20           LD       (ADRS),DE
8009 80
800A E3            21           EX       (SP),HL
800B               22           ;
800B 01 00 1D      23           LD       BC,1D00H
800E ED 79         24           OUT      (C),A
8010 D9            25           EXX
8011 CD            26           DB       0CDH     ; CALL
8012 00 00         27 ADRS:     DW       0000H
8014 01 00 1E      28           LD       BC,1E00H
8017 ED 79         29           OUT      (C),A
8019 C9            30           RET
801A               31
801A 00 00         32 SAVE:     DW       0000H
```

ライブAがドライブCとして使えるようになります。この設定を行ったディスクに先ほど作ったEMM.COMを転送すると自動的にEMMをAドライブとするシステムの出来上がりです。

これによってリブートしたときの時間待ちがなくなって快適な環境が提供されます。ただし，このままだとリセットを押した場合にはEMMにシステムがあってもシステムを転送するので，システムがある場合には転送しないようにするなどの改良を加えればさらに使いやすいものとなるでしょう。各自がんばってみてください。

1989年6月号の質問箱の解答で，BIOS ROMをアクセスするプログラムは8000H以降でないとだめだと書かれていますが，8000H より上位にプログラムがあっても，BIOS をアクセスすることができるようなおいしい方法がないものでしょうか。当然使用機種はX1turboです。　三重県　内山　岳

一応疑似的にBIOSをコールするようなリスト1を作ってみました。ポイントは5005Hです。5002Hで呼んでいるサブルーチンBIOSは最初にSPの内容(スタック)とHLを入れ替えます。つまりHLにスタックに積まれていたデータが入るわけです。ここでいまスタックに積まれているいちばん最後のデータは，このBIOSを呼び出したCALL BIOSの戻り先である5005Hですから HLにも5005Hが格納されます。そして16行から20行でADRSから2バイトにHLの値，すなわち5005Hを格納して，21行でスタックにHLの値を積みます。

このときHLは5007Hですからスタックには5007Hが積まれるところがミソです。23行から29行までがBIOS ROM のサブルーチンを呼び出すルーチンです。最後のRET命令でメインプログラムに戻るときは先ほどスタックに積んだ5007Hをプログラムカウンタにロードして，めでたく5007Hに戻ってくれるというわけです。

このプログラムでは巧み(?)にスタックポインタを使って呼び出しを実現しています。いわばコンピュータをごまかすことによって実現されているものですから，実際にスタックにデータが積まれていく状態を紙に書きながら考えないとわかりづらいかもしれません。このような特殊なプログラムは複数のBIOSのサブルーチンを呼び出す場合などに，CALL BIOSの次に，

　　DW　アドレス

と書くことによってBIOS ROMを呼び出すことができるので便利かもしれません。ROM版のPC-8801用“SWORD”ではこの方法でBASIC ROMのサブルーチンを呼び出していました。しかしひとつか2つのサブルーチンを使うだけなら，こんなややこしいことをしなくても呼び出し位置だけ8000H以降にあるようなプログラムを書いたほうが短いし，わかりやすくすみますので，このようなプログラムを使う機会はごく限られた場合だけでしょう。

今度自分でゲーム用のBGMドライバを作ってみようと思っています。音を鳴らす方法はわかったのですが，OPMに4分音符や8分音符など音長を教える方法がわかりません。どのようにすればよいのでしょうか。　千葉県　喜多　祐一

市販ゲームのようにBGMとゲームを並行に処理しようとするなら，当然ハードウェア割り込みを使うことになります。割り込みをやさしく説明するなら，ある一定の間隔でCPUにいまやっている仕事を休んでもらい，そのあいだに必要な割り込み処理を行って，そのあとさらにCPUに休んだ仕事を再開してもらう，ということになるでしょう。CTCなどを使った割り込みの詳しい解説については，11月号をはじめ以前に何度か本誌で触れられているのでそちらを参照してください。

OPMに直接音符の種類をわからせるような命令はありません。音符の管理は自分自身で行わなくてはならないのです。仮に全音符の長さを256と決めたとすると，2分音符は128，4分音符は64と表せますね。この長さを音長を表すカウンタとして，割り込みがかかるたびにカウンタのワークエリアを1減らし，0になったら音を消してワークに音長カウンタをセットして音を出させ，また0以外の場合はカウンタを1減らすような割り込み処理プログラムを作ります。このようにすれば音符を管理することが可能です。

具体的な流れを追ってみましょう。まず，1チャンネル目のカウンタをデクリメントして0以外ならそのまま，0ならば1チャンネル目用のデータエリアから1音分を鳴らすために必要なデータを取り込み，そのデータに従って処理を振り分けOPMにデータを送ります。そして第2チャンネル以降についても同じ処理を繰り返していくのです。

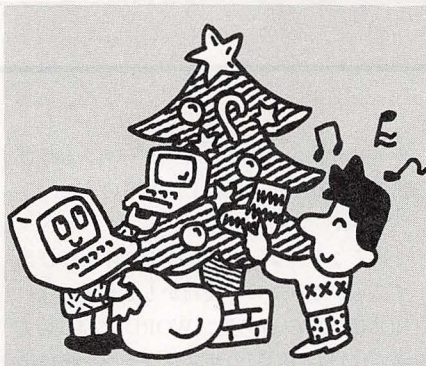
もう，これは一種のインタプリタであるといえます。音源ドライバを作るというのはMML用のインタプリタを作るということなのです。重要なのは効率のよいデータ形式を考えること，できるだけ軽い処理にすることでしょう。

データ転送の部分はどうしようもありませんが，命令によって処理を振り分ける部分はいくらか考慮の余地はあります。たとえば，MusicBASICではそれまで演奏時に行っていた音長計算をデータ格納時に行うようにしたり，Zコマンドなどでデータエリアをまとめたりという工夫が凝らされています。

そのほか，速いパッセージを使わなければ割り込み周期を長くしてCPUの負担を軽減できますし，ゲーム専用ということであれば演奏するデータを特定できるので，データエリアの設定や音長計算など，あらかじめ実行時の処理が軽くなるような設計もできると思います。本誌でも数度に渡って音源ドライバの制作が行われていますのでそれらのソースリストを参考にしてください。　（影山　裕昭）

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を上げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに回答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので，電話番号も明記してくださいね。

宛先：〒102 東京都千代田区
九段南2-3-26井関ビル
㈱日本ソフトバンク出版部
「Oh！X質問箱」係

FROM READERS TO THE EDITOR

クリスマスはやってくるし，Oh!Xはめでたく2周年（創刊7周年半）を迎えたし，X68000は順調に売れているみたいだし……と，12月は明るい話題が多いですね。というわけで，STUDIO Xの1年も無事に暮れてゆくのでした。

◆10月号の「ゲーム面白心理学」はたいへんよかった。特に西川氏のゲーム（ファンタジーゾーン）に対する情熱が読んでいてひしひしと感じられました。僕もこのゲームは絶対買いの2重丸です。このゲームにはとんでもないオマケがついています。まず1面目では右から1番目の基地，2面では？？？というふうに消していき最終面まで行くと，なんと，とんでもない隠し面が……。あとは自分で見て感動してください。　　三浦 裕行（25）神奈川県

◆ファンタジーゾーンのウィンクロンの目玉の上に入ってじっとしていると，だんだん回転が速くなってきてものすごく速くなると実はいつのまにか逆回転になっていて，徐々にスピードが落ちてきます。で，そのうちいったん止まって，また最初と同じ方向に回り出すということを確認しました（暇な奴）。でもボムが当たらないからまったく倒せなかったのでした。

森 啓泰（22）東京都

編集室にある21インチディスプレイがつながったX68000には，昼休みごとに他の部の人たちがファンタジーゾーンをやりにきます。ゲームはほとんど遊ばない人でも，いつの間にか最後まで行くようになるのです。アイデアといい，バランスといい，さすがですねえ。

◆10月号特集のサバッシュについて，戦闘モードは遅くありません。フ○○○○○3，ラ○○○○○○○ン，○○○○○ガーと比べて一番速いと思う。ではなぜ遅いと書かれたか？　たぶん戦闘時のメッセージスピードを80のままプレイしているのではないでしょうか（違ったらスミマセン）。ちなみに私は5でやっています。ほかにも言いたいことがたくさんあります。このソフトは僕の持っている数少ないソフトのなかで最高だと思うだけにくやしい。0.1mmペンで書きまくるぞー！　　石間 崇（19）京都府

（で）君はスピード1でやっているそうです。「比べるものが遅すぎるんじゃ」だって。うーん，彼はせっかちなのかな。

◆サバッシュが終わった。戦闘時間が長く最後の敵なんか30分もかかった。おかげでヨソゴトしていた私はエンディングを見そこねてもう一度戦うはめになってしまった。アートディンクからダブルイーグルのデモディスクが送られてきて，思わず感心してしまった。会員全員に送ったのだろうか？　たいしたものだ。

牧野 智久（21）愛知県

◆サバッシュの採点表のファラナークという項目がとてもよかった（これはやった人にしかわからない）。　　上田 宏美（17）広島県

◆よくぞ「ねじ式」を特集で取り上げてくださった。だいぶ前にNHKで「紅い花」をやったとき以来の大感激。ツァイトの「こだわり」に期待します。　　早坂 栄一（35）岩手県

◆ねじ式の記事を読んだら原作のほうが読みたくなって，本屋を探しまわって「ねじ式・紅い花」を買ってしまいました。スゴイ！　の一言につきますね。とうとうウチにあるマンガも700冊をこえてしまいましたが久々に感動するマンガでした。ゲームのねじ式もやりたいですね。

近藤 英二（18）愛媛県

ねじ式は編集部でも久々に「おおっ」と評判になりましたからね。こういう意外性のあるソフトが増えると本誌のレビューも面白くなるんですが。

◆荻窪氏の言葉「すぐゲームを始めたがる人は，よほど快感原則が強いか，現実がその人にとって耐えがたいものか，のどちらかだろう」……ウムム，三年殺しの秘孔を突かれたような気分だ。暇ができたら，X68000マシン語入門で勉強したことを使って必ずりっぱなプログラムを作るぞ（Cで作ったプログラムは電脳に出してみました）。　　久保田 文彦（28）長野県

◆ジェノサイドに投票したけど，やっぱり一歩間違うとクソゲーかもしれない。はっきりいってカタイ。カタすぎる。ゲームをしていてスカッとしない。ある程度，スカッとするザコキャラが欲しかった。それから，ボスキャラにはゲージを表示してほしい。誰でもラスト近くまで行けるようなバランスとマニアにはVery Highモードを付けるとかして対応してもらいたい。

岡山 稔明（22）長野県

X68000ユーザーを魅了した迫真のパワーをたたえる声もあれば，ゲームデザインに問題ありと評する声もあり。話題作ですねー。

◆僕は今98ユーザーですが，本当はただのゲーム好きなのです。それがなぜか98を買ってしまって今に至るというわけで……。だから，Oh!Xを読んでX68000も結構いいなと思い，次に新しいのが出たら，なんとかX68000を買いたいと思います（98売ってね！）。

柿平 貴司（16）東京都

またひとつ98が死んだ　……。

◆中古ソフトを買いだめしてしまった（安いからついつい手を出した）。Willやテグザー，ロストパワーに，はぁりぃふぉっくす2，暗闇の視点，etc.　ああ，なつかしー。思わずタイムトラベラー。そしてもーすぐ国試がある。なんもやってねー。すごく，やばい！

大津 和之（19）福岡県

そのあたりのゲームソフトって今では貴重ですよね。東京だってなかなか手に入りませんよ。あっそうそう，どなたか存じませんがボンバーマン送ってくれた方，ありがとうございます。

◀大村 直人（17）北海道
わざわざ珍しい飲み物を届けてくれてありがとう。だまってピーチアップル飲んじゃったやつは？　誰だぁ，んっ？

◀味野 真一　群馬県
ほんと，もうすぐクリスマスですよねえ。素敵なイラストありがとう。一瞬，霧降高原を思い出しちゃいましたよ（え，知らない？）。

◀大野　真実　静岡県
またまた登場の大野サン。今度はA-JAXできましたか。しかし，1円玉ならぬ1円弾の嵐とはいいところに目をつけましたね。

◀丸藤　俊之　神奈川県
これまた登場の丸藤サン。こちらはまっとうにデザインセンスで攻めてきますね。ちなみに，最近ファンロードも荒らしてませんか？

◆X-BASICプログラミング調理実習でチェスボードを作っていましたが，僕はチェスを知らない。できれば，11月号でチェスのルールを教えてください。　伊藤　健一（17）兵庫県

今月はタートルグラフィックの解説があるんだけど，きっと「僕は数学がわかりません……」というハガキがくるだろうな。どうしましょ。

◆日本橋は恐ろしいところです。X1Gmodel 10（新品）が￥1,000で売られておりました。同じ店のシャープのジョイカードよりも安いのはなぜだろう。　荻野　欣也（20）兵庫県

あ，あんまりですね。ところで，Gのモデル10って最後のカセット内蔵コンピュータなんじゃないだろうか。

◆Oh!Xの記事は笑わせてくれるだけでなく，思いっきり考えさせてくれるようなもの（10月号でいえば知能機械概論）もあってとてもいいと思う。本文の言葉を借りれば，自分はサブリミナルメソッドされているかもしれないという不安感は，なんともいえないものがある（こういうのはほかでは味わえない）。
大森　幹雄（17）神奈川県

Oh!Xには深層心理に訴える数々のメッセージがいたるところに仕掛けられているんですよ。ふぉほほほ（ハッタリだってば）。

◆ついに念願のX68000を買いました。最近，僕の部屋は完全にゲーセンと化しています。これではまずいと思い，ふと気づいたのですが，X68000を使おうにも何ひとつ知らないと……。だからOh!XではもっとX68000を使うために必要なプログラムを載せるべきです。
久下沼　信（20）石川県

◆エロエムエッサイム，エロエムエッサイム。我は求め訴えり。出よ，
第13使徒　レーザープリンタ！
第14使徒　光磁気ディスク！
第15使徒　DSPボード！
第16使徒　68030ボード！
瀬戸　暢彦（23）神奈川県

あやしげな呪文だ。効くといいけど……

◆スタッフ希望でしたが，なんです！　あの自由論文6000字って!!　4月号のときはレポート用紙2枚だったのに……。悲しくてしようがありません。助けてください。
佐藤　久彰（20）茨城県

6000字って一瞬多いように思うけど，実は記事でいうとたった2ページ分ですよ。本誌のスタッフなら一晩仕事かも。

◆X1のソフトが発売されなくなってしまった。しかし，X1を手放すのはまだはやいぞ。X1はプログラムを楽しむ（苦しむ？）ための道具としてユーザーの元に帰ってきたのだ。今こそ，X1に全力を投入する絶好の時期だ！　と私は思う（だってX68000買うお金がない）。
田中　啓介（23）神奈川県

◆デービーソフトさんがX68000用フラッピー2を登場させてくれますね。こうなったら，次はタイトーさんが，あの名作「ちゃっくんぽっぷ for X68000」を出してくれることを期待しませう。　相原　弘一（19）大阪府

◆X-BASICもいいのですが，BASIC98のようなハイスピードなBASIC，そしてコマンドがHuBASICと変わらないようなBASICは出ないのでしょうか？　X1turboで作ったBASICプログラムをハイスピードでX68000で動かせたらいいなあと思う毎日です。　泉　正寛（31）滋賀県

もとX1/turboユーザーならやっぱりそう思いますよね。だれか作ってくれないかな？

◆映画「ノーライフキング」の学習塾では，X68000が使われているようですね。やっぱり絵になるのは98よりもX68000ってことでしょうか。目立ちたがり屋のTOWNS君が出てこないのも不思議だ。　沼部　栄士（20）群馬県

まこと君のお母さんは98を使っていましたけどね。TOWNS君は出てこなかったなあ。

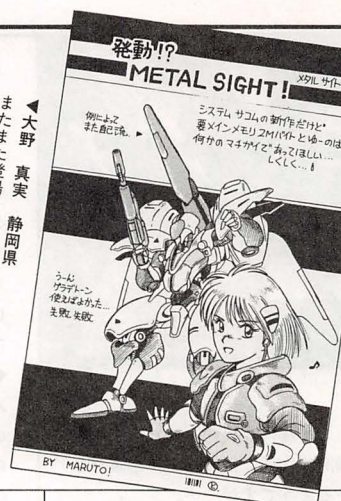
◆コンピュータアニメーションのVol.1,2をLDで観ました。いやぁすごくよかった。実写としか思えないようなものあり，シミュレーションあり，コミカルなものあり，と内容も盛りだくさんでした。まだ観ていない人には，ぜひとも，と思います。　大塚　京吾（20）岐阜県

◆XCのバージョンアップがあるって聞きましたが，いつごろなんでしょうか？
岸　雅樹（20）大阪府

着々と進んでいると思いますよ。でも，一太郎のように毎年バージョンアップというわけにはいきませんからねえ。

◆サポートツールStationery PRO-68Kの登場で，電子手帳はリッパなX68000の周辺機器となった。10月からはICカードにゲームなど新しいものも発売されるようだし，Oh!Xでも電子手帳の活用記事を載せてください（あの小さな液晶でどうやってマージャンをやるのでしょう。テトリスも出るみたいだし）。
田中　正志（21）千葉県

今度はサイバーノートというのがデータショウに出ていましたね。面白くなりそう。

◆つい数日前知ったことですが，今年のクラスのなかにX68000ユーザーが私のほかに2人もいました。やはりOh!Xは買っているとのことです。なんだかうれしくなってしまいました。
遊佐　勝（17）埼玉県

◆僕のハガキが載っていたことに2カ月も気づかなかった。さらに，読みはじめてから5年近くになって，STUDIO Xにペンネームなるものが存在しないことに初めて気がついた。妙に感動した。はあ～。　河辺　義信（17）愛知県

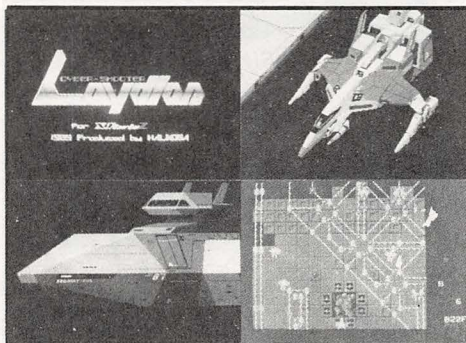
◆最近のシャープに対して一言。「あんたらEXE会員を何だと思ってんだい！　EXEリーダーズカップの宣伝ばかりして，ユーザーにばかり頼らず少しは自分でX68000の宣伝をしたらどうだい！」。　松永　正弘（19）千葉県

まあ，某社くらい宣伝費があったらマシンを安くしたほうが売れると思うけどな。でも，X68000ユーザーの士気と野望（なにそれ？）を煽るためにもシャープさんには宣伝に力をいれてもらいたいですね。

◆見一つっけ。ツインビーのサンプリング音が聴けるぞ。COMMAND.Xを立ち上げDIRを見ると～pcm.～というのがある。でもってCOPY ～pcm.～ PCMとすると聴けるのだ。おそろしいことに面カウントの声がえんえんとあるのにはまいった。ツインビーには終わりはないのか？　それともひとつ“キャハハ”という笑い声はゲームに出てくるのだろうか？
小川　貴弘（24）三重県

◆X68000が発表されてから3年が過ぎようとしています。世間ではLAPTOP版X68000やX68030のウワサで盛り上がっていますが，まず開発環境のさらなる整備（XC Ver.2とソースレベルデバッガとmakeはどうした！）とビジネスアプリケーションの充実（早くまともなワープロ出して！）も忘れないでほしいですね。あっそうそう，VS.X Ver.2も早く出してね。
田中　義彦（26）東京都

カラー液晶のラップトップは高すぎてほとんど売れていないようです。やはりあせらずに，環境整備に力を入れてくれたほうが結果的にいいですよね。

◆X68000に光磁気ディスクをつけてX68000 EXPERT-MOとかX68000のラップトップを出してほしいけど高すぎるとちょっと！　今はそれよりソフトやボード類を充実させてほしいと思い

ました。　　　　藤代　茂明（18）神奈川県

祝さんの満開製作所では光磁気ディスクを導入しているようですが……。

◆10月号166ページの原秀樹さん。あなたの言っていることは本当ですか。世界初のZ専用シューティングゲームですって？　早く完成させて，早く，早く，早く〜。Z専用かぁ〜。Z専用，Z専用。これはいいや。Zユーザーにとっては神の声だよ。絶対に完成させてくださいよ。

大島　竜次（19）愛知県

実は原秀樹さんから写真が届いているんで，4点セットで載せちゃいましょう(上)。まだ，サンプルを見せられるような段階ではないそうですが，リストを載せられるような大きさじゃないようですね。どうしましょう。ね，原さん？

◆アドバンスト・ダンジョンズ＆ドラゴンズの原作となったシナリオは，ゲームブックの『魔都の黒竜』でなく，ドラゴンズ戦記第1巻『魔都の黒竜』である。間違えないでほしい。

井上　敬介（18）神奈川県

失礼しました。すんまへん（へこへこ〜）。

◆「失敗する主な原因は一気に大きなプログラムを作ろうとしてしまうからである」。このBondingの華門さんの言葉には反省させられた。これまで，アイデアだけで挫折してしまったことが何度もあった。でもですね，気合いが入るとやっぱりすげープログラム書こうって気がはしりますよ。　　　今井田　和也（17）愛知県

◆私は女の子がOh!Xを買っているところを今まで見たことがなかった。しかし，ついに目撃してしまった。　　　遠藤　亮司（19）栃木県

最近，お隣のOh!FMには女性スタッフの方がよくおいでになるんですよ。まったく羨ましい。

◆ついにX68000を購入しました。「やっと自分のコンピュータにめぐりあった」という感じです。これからは，Oh!PCから乗り換えますんでヨロシク。私のPC-286US-M20は友人の家に養子に行ってしまいました。元気でね。いつでも帰ってきていいんだよ！　黒沼　納樹（20）埼玉県

◆私は仕事がら思考がCOBOLしているのです。X68000にCOBOLコンパイラが出る予定はないのですか？　もし出たら気合いを入れてEXPERT-HDを買うのに〜。それからもうひとつ。最近CPUを何台も持っている人が増えたようですが，ディスプレイも何台もあるのでしょうか。私の部屋はそんな余裕はないのでディスプレイ1台に本体を何台もつなげられるセレクターの作り方をぜひ教えてください。

倉持　悟（24）東京都

編集室でもCZ-600DにX68000（アナログ）とX1turbo（デジタル）をつないで正面のアナログ/デジタルスイッチで交互に切り替えて使っています。便利ですよ。

◆草の根ネットのホストパソコンは圧倒的に98が多いと思うが，少なからずX68000のネットもある。X68000のネットは98のネットに比べると明らかになにかが違う。X68000のネットにはパソコンのパソコンによるパソコンのためのネットが成り立っている。PDSのダウンのことばかり考えているような人はあまりいない。

遠藤　勇（32）大阪府

◆そのうちパソコン通信のことやってほしいです。まったくわからないんです。NTTに怒ってばっかりいないでさ，やってみてくださいよ。

高橋　政秀（16）東京都

そうですね，そろそろパソコン通信の話もやってみたいのですが。

◆近い将来，Oh!Xはソフトの出ないパソコンユーザーがボートピープルよろしく集まるかもしれません。しかしOh!Xは資格審査なんか絶対しないでくださいね（なんのこっちゃ）。

斉藤　修（21）宮城県

いくらうちでも，斉藤由貴や宮沢りえの踏み絵までは……，えっ，そんなこと聞いてないって？

◆Oh!PCから88が締め出されてしまいました。どうか，Oh!Xで88ユーザーを引き取ってください。表紙のタイトルの上にPC-8801の名を加えてくれるだけでいいんですがダメでしょうか？　最近，X1turboZユーザーの友人がPC-286VFを買い，PC-8801(初代)ユーザーの私はX68000 PROを買いました。さて，2人のコンピュータを見る目は正しいのでしょうか。

宮崎　圭介（21）福岡県

少なくとも，あなたの目は正しい，まったく正しい，正しいなあ。

◆僕は，パソコンに求められるあらゆるアート機能を取り去った驚異の80×50ドット8色フルカラーグラフィック，ファミコンにも劣る3オクターブ単音演奏のMZ-700ユーザーです。久しぶりにOh!MZ（X）を買ってしまった。

黒崎　亨（16）富山県

本格的なCGの世界では16777216色のことをフルカラーと呼ぶようですが，パソコンでフルカラーといったらやっぱり8色ですよね。えっ，違う？

◆MZ-1500に愛の手を！　猫の手でもいい。

船越　直弥（17）北海道

◆CGのコンテストとか，Musicのコンテストとか企画すればいい。たぶん皆さん望んでいると思う。　　　菊田　俊彦（31）茨城県

◆PCエンジンSGに一言。“PCエンジンVAとでも名前変えたら？”以上，深い意味はありません。

工藤　隆（19）埼玉県

◆僕の趣味はプログラムをバシバシ打ち込み，いじくりまわして楽しむことです。だからOh!Xだけが頼りです。　　　小塚　紀義（18）宮城県

◆わが高校の数学準備室には，SuperMZ（V2）＋MZ-1P17のセットが5台もあります。一応，数学同好会（生徒はおろか先生も知らなかった）に入っていて，したいほーだいにいじれます。はっきりいって，SuperMZが5台もドライブ音を立てている姿は異常です。

吉沢　毅（17）福島県

サイバーだなあ。なんのこっちゃ。

◆あー，ついに買ってしまったX68000。本当は68Kボードを自作するつもりだったけど，インターフェース誌に「やはりホストマシンがないとね……」というのがあり，う〜んと悩んで，結局2カ月分の給料が化けてしまったというわけです。こいつなら，うちの大学のクラブでもメイン機種だもんな。買いに行ったときは，頭金がそのまま消費税分で，一瞬のうちにサイフが軽くなり，その後なにも買えなくなったのはつらかった。しかし，これからはバリバリやで。

播戸　行徳（21）兵庫県

これまた，ストロングなユーザーになりそうな方ですね。

◆最近，受験ネタが少ないですね。そういう僕はいま，高校3年の現役バリバリです。推薦入試を受けようと思って目下勉強中です。いまは

◀荻田　俊平（18）和歌山県
サイキック少女団？　あぶないタイトルだなぁ。ところでフロッピーを送るにはダンボールや厚紙に挟んであればべつに問題ないですよ。

◀高橋　弘幸　神奈川県
そうそう，Oh!Xになってもう2年なんですね。料理されたX68000にチャイナドレスのおねーさん。なにかを象徴しているみたいですね。

ふんぎりをつけるためX68000は箱にしまってあります。来年は情報系に入って，ピカピカの大学生として投稿したいです。

漆畑　幹雄（17）静岡県

リクエストに応じて，というわけではないんですが……。

◆ああ～，早く「フラッピー2」がやりたい！　早く「遥かなるオーガスタ」がやりたい！　早く「ローグアライアンス」がやりたい！　早く「スーパーハングオン」がやりたい！　あ～，早く「サンダーブレード」がやりたい！　でもその前に，早く「ぢゅけんせい」を終わりたい!!

国政　寛（18）大阪府

◆X68000が欲しくて，ジェノサイドがやりたくて，MIDIで音楽したくて，65536色でお絵かきしたくて，アフバナとA-JAXで目を回したいのに，受験の真っ最中で，金がなくて，家が狭くて，偏差値がなくって，問題集の54番が解けない！　明日からYAWARA!がTVで始まる。その前に今夜からCITY HUNTER3がTVで始まってしまう。ああ，どうしたらいいんだ！　それもこれもOh!XがTVCMしないからだ。なぜOh!PCはTVCMしてるんだ！　西畑　義彦（18）大阪府

一方的なマシンガントーク（エディ・マーフィーか？）。思わず，早口で読んでしまうじゃないですか。

◆9月号の相賀さんへ。実はシャープより先に，ザイ◯グ，日本サ◯ライズ，そしてバ◯ダイが共同で，64ビットCPUを開発中という根も葉もないウワサを聞いた。その名もZZ-0080(ダブルゼータ，ダブルオウエイティと読む）という。Z80の後継は確実？　宮原　大（16）長野県

なんというか，異様なリアリティを感じてしまうなあ。

▲岩本　智雄（18）埼玉県

こちらも2周年のお祝い。いや～感激だなぁ，セーラー服の女の子にバラの花束なんてもらったの初めてだもの。うるうる……。

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目（売る・買う・氏名・年齢・連絡方法……）を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集室では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。

仲間

★わが「DRAGON SYSTEM」では，さらなる会員増を目指しています(現在約50名)。パソコン，メガドライブ，テーブルトークと，ジャンルにとらわれないファンタジーRPGサークルで，グループSNE公認です。月1回の会誌発行が主な活動（すでに3号まで。オフセット）で，今回会員が増えたらフルカラーになる予定です。興味を持たれた方は，以下の住所に62円切手を同封して手紙をください。（文責：岸水明）〒651-11 兵庫県神戸市北区君影町1-12-104 横内隆（17）

★「Gamers Club68K」というサークルを運営しています。機種はX68000を持っていてゲームの好きな方なら誰でもけっこう，内容は月1回の会誌発行，その他ゲーム大会などいろいろです。詳しいことは62円切手同封のうえ封書にて。〒761-06 香川県木田郡三木町氷上4168　小西茂（16）

★X68000ユーザーを対象とした「ゲーマーズX68K」を発足させるにあたり，ゲームが大好きで「ゲームならまかせろ！」という人々を大募集しています。内容は月1回の会報で情報交換をしたいと思っています。まずは62円切手同封のうえ連絡を。〒260 千葉県千葉市真砂3-10-11 鈴木淳（16）

★X68000ユーザーを対象とするクラブの発足にあたり会員を募集します。情報やPDSの交換をしたいと考えています。初心者の方，大歓迎です。詳しくは62円切手同封のうえ連絡を。〒793 愛媛県西条市大町397-11　片木章人（16）

★「TRACK・DOWN」では新規会員を募集します。X1のディスクユーザー（turboを除く），MSX2，88SRユーザーで興味のある人は62円切手同封のうえ連絡ください。活動内容は中古ソフト，ハードの交換，共同ソフト開発（X1のみ）です。ソフトを作りたい人はぜひ会員になってください。〒028-05 岩手県遠野市早瀬町2-4-11 松田義徳（17）

売ります

★FDD・MZ-1F07(ケーブルつき)を4万5千円で。連絡はハガキで。〒182 東京都調布市深大寺元町1-8-23 新調布ハイツB-103　白井英樹（20）

★MZ-700用のQDドライブMZ-1F11＋インタフェイスMZ-1E14を合わせて2万5千円で。PCG-700を1万円で。共に完動，箱，マニュアル，付属品あり。送料込みで。連絡は往復ハガキで。〒171 東京都豊島区南長崎4-2-13 堀方　細谷晴夫（16）

★X1用漢字ROMボードCZ-8KRを500円(送料込み)で。完動品です。〒444 愛知県岡崎市田町12-1 千賀知之（41）

★ドットプリンタCZ-8PD3を5千円で，X1turbo用2D/2HDドライブCZ-520Fを3万円で。連絡は往復ハガキで。〒182 東京都調布市小島町3-21-18 第二本多荘208号室　池田健一（22）

★ロジテックのLHD-34V（X68000でもそのまま使えました。ほぼ新品，完動，美品，付属品全てあります）。送料込みで9万円で。なお，希望するなら新品の感熱紙（100枚入り）もつけます。詳しくは往復ハガキなどで。〒767 香川県三豊郡高瀬町新町駅西1433－5　関康弘（18）

買います

★X68000用1MB増設RAMボード（CZ-6BE1A）を1万6千円前後，MIDIボード（CZ-6BM1）とMT-32をセットで4万5千円前後，MT-32だけでも可。完動・付属品・説明書つきならば多少の傷や外箱なしでも可。連絡は希望価格・状態を明記のうえ往復ハガキにて。〒358 埼玉県入間市宮前町6-1　加藤繁明（20）

★X68000用1MB増設RAMボード（CZ-6BE1）を1万5千～2万円で。完動・付属品つきならばその他は問いません。連絡は希望価格・状態を明記のうえ往復ハガキにて。〒670 兵庫県姫路市西今宿2-6-16　林健太郎（16）

★MZ-2500増設ビデオRAM（MZ-1R27（A），またはコンパチ品），MZ-1P17用第2水準漢字メモリ（MZ-1R29（A））を，それぞれ6千～7千円で。希望価格を明記して，往復ハガキで連絡してください。〒331 埼玉県大宮市日進町1-585-4　今野和浩（18）

★X1用の漢字ROM（CZ-8BK2）を送料込み5千円で。無改造・完動品に限る。連絡は往復ハガキで。〒284 千葉県四街道市大日27-40　浅野一行（17）

★X1用プリンタCZ-8PC2，CZ-8PC3を2万円前後，FM音源，カラーイメージボード，320KB外部メモリを1万円前後，第2水準漢字ROM，マウスを4千円前後で。連絡はハガキで。〒598 大阪府泉佐野市上瓦屋186-2　松浪勝義（19）

★X1用RS-232C・マウスボードCZ-8BM2を8千円で。ノーマルケーブルまたはマウスつきなら1万円。連絡は往復ハガキで。〒921 石川県金沢市窪1-102 まりみ寮23号　東田貫司（19）

バックナンバー

★Oh!MZ1987年8月号を送料込み1600円で。すべての内容がわかれば多少の破損，汚れ可。連絡はハガキで。助けるつもりでお早めに。〒774 徳島県阿南市上中町岡234　牧岡孝則

編集室から from E·D·I·T·O·R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今回は，10月号の記事に関するレポートです。

●X-BASICプログラミング調理実習は，話の進め方が「機能の分析→部品作成→統合」のステップを一応踏んでおり，スマートに理解をうながすことと思います。それにしてもX-BASICの外字の取り込みって便利なんですね。直接プログラムに組み込めるとは……。X68000のない自分にもかなり参考になる点があり目からウロコが落ちた感があります。
中野賢一（29）MZ-2000，X1/G/turboII，FM-8，FP-200，PC-8801/9801，PC-1251，B16LX　山口県

●カードゲームBondingは，プログラムリストが短いわりに結構おもしろい。アイデアですよね！　プログラムがBASICで書いてあるので，比較的初心者でも改造がしやすいというのは大いなるメリットだと思います。特に本文中でも，どのへんを改良したらいいのかも書いてあって，初心者が打ち込むには適したプログラムといえます。さすがは華門さんと感心してしまった。私の場合，1時間強でリストを打ち込みましたが，結構いろいろと遊べていいと思います。これくらいまでのプログラムが「ショートプロ」などにどんどん載れば楽しいと思います。
森川一（24）X1turboII，X68000ACE-HD，PC-286LE-STD　北海道

●昔，コンピュータグラフィックがはやったころ，みんな方眼紙をもって座標を読み取り，LINE文を使って絵を描いたものですが，MMLのリストを載せるLIVE in '89には，どうもあのころの「手書きでドットを読む」という作業を思い起こさせます。いままでの貴誌の特集では，エディタなども発表されていますが，むしろ「エディタを使って曲をつくり，それをいかに効率よくMMLに変換していくか」という議論がそろそろなされてもよいと思います。
湯沢聡（26）MZ-2500/2861，X1turboIII，X68000，MSX，PC-6600，PC-1360K　東京都

●時代の流れか，記事の主流はX68000であるが，X68000マシン語プログラミングを読むと，かつての主役であった8ビットに触れてきたであろう筆者の何かしらの心が見えてくる。内容的には「フィルタとはなんじゃらほい」と浅学な私にもなかなかに読ませてもらえるものだった。こういった，誰にでも通じるであろうテーマの記事はいいものである。
大津和之（19）X1turboZ　福岡県

●祝氏のエディタはなかなか面白いものになってきた。ソースリスト中に注釈文が多いので自分でいじくることができてよい。いじっているうちにCでの組み方がわかってくる。完成が楽しみだ。
西田宗千佳（18）X1F，X68000　福井県

●THE SOFTOUCHでは，ツールソフトなどの紹介はかなり充実していてよいと思うのですが，ゲームソフトについてはもっと単刀直入に，このゲームのここがよい，ここが悪いとハッキリ書いてほしいと思います。あくまで客観的にゲームを評価してもらいたいです。
藤田康一（18）X68000PRO　静岡県

●いままでLIVE in '89のリストは長すぎて打ち込む気がしませんでしたが，OPMA対応のトップランディングはリストが短そうだったので打ち込んでみました。曲はドラムがADPCMのおかげで思ってた以上にすごかったのでおどろきました。これくらいの長さでしたら1時間程度で打ちおわりますので，できればこれからもこの程度の長さのプログラムも載せてほしいと思います。また，特集の「ガウディ・バルセロナの風」のレビューでは，HRシステムやOPSなど，アルファベットからは何かさっぱりわからないところを丁寧に解説してあってよかったと思います。評価できるところは大いに評価するけれども不満な点もちゃんと指摘してあり，これからこのソフトを買おうとしている人にも参考になったのではないでしょうか。
田中実（19）X1turboII，X68000ACE　大阪府

●なんか今月の荻窪氏の記事はずいぶんと難しいテーマばかりのような……。ただ，満足と快感はかなり違うと思うんですよね。前者はあることを行った結果であるし，後者はあることを行っている最中に得られるものなんですよ。私がゲームをしているときは，両者が一致しないほうが多いですね。快楽について私が思うのは，最終目標に至るまでの行程で，いくつかの壁を越えた一瞬をさすのじゃないでしょうか。うーん，むつかしい！
藤原博人（25）X1turboZ　鳥取県

●圧力で変位を感じるサイバースティック。これがあれば確かにいままでの10倍以上の快楽を感じることができるだろう。同感，同感。いままで，なぜこのゲームは面白いのかと考えたことは多々あったが，それに対する答えがいろいろと書かれていて面白く読めた。
大山栄一（16）X68000ACE-HD　大阪府

ごめんなさいのコーナー

11月号　いまどきの32ビットCPU
P.47　マイクロプログラムの考案者名が誤っていました。正しくはWilkes(ウィルケス)です。

11月号　マシン語カクテル
P.116，117　MZ-700のスクロールプログラムで画面桁数が誤っていました。

LD BC，79　→　LD BC，39

に訂正してください。
また，P.116の右段18行目の（テキスト）と（アトリビュート）は順番が逆になっていました。正しくは，

D800H－D000H＝800H
（アトリビュート）（テキスト）

となります。

11月号　LIVE in '89
P.140　オブ・ラ・ディ，オブ・ラ・ダで前奏部分の音が1音違いました。

240p(1)＝"f8fff8fff8e8e－8d8"

に変更してください。メタルホークで旋律の一部が1音違っていました。4260行の2ブロック目"e－&d&e－&d&c&b……"を"e－&d&e－&d&c>b……"に変更してください。

11月号　MZ-2500グラフィックエディタ作成講座
P.123　印刷ウィンドウでMZ-1P17用の色設定に不備がありました。

FE06H　20 01 7B 3D 20 02 7A C9 79 C9
→　28 05 3D 20 03 7A C9 4B 79 C9

に変更してください。また，10月号ごめんなさいのコーナーで訂正部分のアドレスが誤っていました。

DEEF　→　DFEF
E030　→　E060

に訂正してください。

バグに関するお問い合わせは
☎03(230)7683(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。
また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

足かけ2年のC言語特集だ

▼今月の特集では久し振りにC言語を取り上げました。本誌では読者の7割近くがX68000ユーザーで，そのうち5人に1人がXCを持っているようです。ここらでじっくりとCの基礎からやってみようということになったわけですが，いかがだったでしょうか？　今月号ではC言語の基礎となる部分を地道に解説しましたが，来月は今回の続きとしてより応用的なプログラミング記事を用意してあります。ただ「うーん，やはり初心者にはCはむずかしい」と挫折しそうな方もいらっしゃるかもしれませんね。そこで，もっと気軽にC言語の世界をのぞいてみたいという人に，やさしい参考書をご紹介しましょう。Oh!PCに以前連載された林晴彦氏の『Play the C』(上下，各1,500円）が初心者にもわかった気にさせてくれる読みやすい本です。プログラムは特に組めなくてもいいという人にはおすすめです。なお，「C調言語講座PRO-68K」は都合により残念ながらお休みです。また次号をお楽しみに。

▼最近，「X68000でプログラムを組んだのですが大きすぎて雑誌には載せられないものはどうすればいいのかわからない」といった声を聞きます。編集部ではそうしたプログラムの扱いをどうすればいいか検討していますが，読者の皆さんのご意見もお聞かせいただけたらと思っています。Oh!Xで商品化する，フロッピーを付録にする，某DISKマガジンに売り込む……など，皆さんのアイデアをお待ちしています。

さて，次号は恒例のGAME OF THE YEARのノミネート発表です。そしてもうひとつ，本誌始まって以来の特別別冊付録「X68000ゲームソフトカタログ」が付いてしまうのです。これはゲームファンならずとも必携の保存版アイテムとなるでしょう。期待してください。

▼本誌宛の年賀イラスト，クリスマスカード，の発表は3月号で行います。今年はCGに期待したいですね(フロッピーでも可)。皆さんよろしくお願いします。ではまた来月。

投稿応募要領

- 原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討の上，製作したハードが必要な場合はご連絡いたします。
- 投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒102 東京都千代田区九段南2-3-26井関ビル
日本ソフトバンク出版部
Oh!X「㋢㋮㊔」係

SHIFT・BREAK

▶去年の今頃，半年ぶりに出した靴をはいた。そのときはなんとも思わなかったが，脱いでふと足の裏を見ると茶色いパリパリしたものがくっついている。「なんだ，これ？」靴の中を見ると，押し花のようなものが……。「にゅぎょーっ！　ゴキブリのミイラだーっ！」以来ゴキブリに恐怖と憎悪を抱いてます。
（何故唐突に思い出したかは次号，H.U.）

▶これが載るころにはちょっと時期はずれの話題になってしまうが，10月下旬は学園祭シーズンで非常に忙しい。プログラムはあるわ，レポートはあるわ，試験はあるわ，あげくのはてに編集後記はあるわで困ってしまう。そんなこんなで，結局98の天下統一はおもしろくてハマってしまうのであるという言い訳をしておこう。（亀）

▶三共のリゲインのCMは時任三郎のスーツに身を固めたビジネスマンスタイルが周りの雰囲気とミスマッチしているところが楽しくて，僕の好きなCMの1つである。なんでもあのCMで使われている歌がCMなんとかコンクールの歌部門で表彰されたそうだが……。アリナミンVとリゲインではどちらが売れているのでしょうか。（H.K.）

▶おそらく今日，11月某日にOh!Xのスタッフは飲み会に行くと思います（ただし，編集部非公認です念のため)。さらにおそらくは某氏の家に行って麻雀大会，テトリス大会，フイッシング大会，MONOPOLY大会，バクロ大会，そして早朝ぶっちぎりドライブ大会へとなだれ込みそうです。ああ，恐ろしきはZ80's Bar。（S.K.）

▶昔，大学で，クラスの試験対策用に講義の解説書を結構気合い入れて書いたことがあった。数年後の今でもなぜか出回っていたらしく，それを読んだという内容のハガキを，ある読者が寄せてくれていた。自分の書いたものに反響があるというのは，それをどこかで誰かが読んでくれているということで，物書きにとっては実に気分のいいものだ。（A.T.）

▶えーと，あの話はしたっけ？　ダッシュ一番，自転車で車を追いかけ坂を登っていったおまわりさんの話。してない？　じゃ，在学中，毎年新入生に間違えられ，いまでも大学の前を通り過ぎようとするとサークルに誘われる男の話は？　電車の中でギターを掻き鳴らし足を踏み鳴らしてラクカラチャを歌うおねーさんの話とか？　それから……（Mu）

▶ところで，思考と感情を分離することはできるだろうか。昨日と明日。どちらが確実なのだろうか。未加工で読みにくい1次情報と，加工されて脳にやさしい2次情報と，あなたはどちらを選ぶだろうか。押しつぶされて血反吐を吐きながら自由でいるのと，快適で安全な環境と引き換えに奴隷でいるのと，どちらが人間らしいだろうか。教えておくれ。（K）

▶JUNETのニュースより。F社のある人が自社のTRONチップにGNUのCを移植したところ，現存のTRONチップのどのCコンパイラよりも高性能だったそうだ。GCCの偉大さを再認識するとともに，それを越えられないF社（あるいはOEMかな）の技術力を哀れに思った。あの健ちゃんが見たらなんと思うだろう。（右と左のわからないKO）

▶今年こそ革のコートがほしい，と渋谷へ買いに行ったときのこと。「彼女，18？」と店員さんに聞かれ「24です」と言おうとした矢先，「ごめん，20ぐらいか」と勝手に決めつけられてしまった。そういえば，前に新宿で遊んでたときも高校生に間違えられて補導されかけたし……。なんか若く見られすぎて，嬉しいのを通り越して悲しいものがある。（E.O.）

▶もうすぐ1980年代も終わりです。この10年間，私は何をやってきたのでしょうか。これから10年間，何かいいことはあるのでしょうか。と，そんなことを考えてしまうほど，今月は忙しかった。はっきり言ってもう頭が腎虚。などと逃避していたら横から編集長が「来月は年末進行だからもっと忙しいよ」と，暖かい励ましのお言葉。（S）

▶机の上に3台目のコンピュータがきた。20MHz68030，8MバイトRAM，256MバイトHDDとそこそこ強力。向かいの机には16MHz80386のマシンがあるが，ま，格が違う。はずなのだがウィンドウは重い。やっぱRAMは16Mバイトいるなぁ。X68000も初代からEXPERT HDに変わって1カ月半。SRAMを見ると，起動回数44回，平均起動時間12.7時間……。（U）

▶この時期（12月号の編集が終わるころ）になるときまって，「そういえば年賀状やクリスマスカードの作成講座をやるんじゃなかったっけ？」「もう手遅れですよ」というやりとりがあり，「こんど暑中見舞いの時期に合わせてさあ……」といいつつまた年末を迎えることになるのです。なさけない。
（readers'ぎゃらりぃ年4回をめざすT）

microOdyssey

色のついた夢を見るということは（控え目にいっても），あまりよくないことのようにいわれてきたように思う（最近は少し変わったのかもしれないが）。しかし，友達と話してみると，結構，色つきの夢を見ている者も多い。

私自身も，ある時期から天然色の夢を見られるようになった（いつでもというわけではない)。よく考えてみると，それは色彩に対する認識が強くなった時期からだということがわかる。学生時代，少々絵を描いていたことからか，色に関する認識が少しばかり変わったらしい（先ほどの友人たちにしても，ほとんどが絵を描く人間だった)。

小学校の図工の時間には，なにかほしい色があったとしても，絵の具を混ぜあわせてその望みの色を作り出すということはほとんど不可能に等しかった。綺麗な青みがかった緑がほしいのに，緑に青を混ぜていくとどんどん濁った色になっていき，収拾がつかなくなってしまった，という記憶は誰にもあるのではないだろうか。

慣れるにつれて，次第に好みの色彩が作れるようになってくる。それはたぶん，日常目にするものに対して，「これは○○色」という意識を持つようになるということだ。目から受け取ったイメージだけでなく，自分からその色を定義していくようになって初めて，色彩を認識したといえるのかもしれない。

夢の中でもこれと同じことを行う，すると自然と色がついてくる。夢の中でもちゃんと色彩のイメージがあれば，それを感じとれるはずなのだ。

「あらゆるものには色があるものだ」というのは人間にとって実に自然な感覚のように思われる。奇跡の人として知られる，ヘレン・ケラーは盲人でも色彩に対する理解を持ちうることを強調したうえで，「私に色がわかるかどうかは別として，ひとつの統一された世界を形成するためには色があることが必要なのです」と語る(『私の世界』)。盲人であるヘレン・ケラーが色彩に関する理解を得るにいたった過程には，知識による類推と，潜在的記憶による霊感があったという。事物を理解するとき香りや色彩を伴っていたほうが，より相互理解が得られる。よって，人にはそれらを潜在的に感じる力を持つのではないか，というのだ。

ものにはなんらかの色があるとするほうが，自然な感覚として受け入れやすいにもかかわらずふつうの夢には色がないとされるのは，単に色彩に対する認識の問題にすぎないのではないだろうか。パソコンもアナログRGBが当然のようになり，私たちのまわりは色彩であふれている。もしかしたら，カラーテレビ世代にとっては「なぜ，色がないのか？」ということのほうが不思議なことなのかもしれない。

さて，最近見た夢でいちばん変わっていたのは，夢の中で以前見た夢をもう一度見ているという夢だ。夢の中の夢に現れているのは「私」でそれをわりと冷静に見守っている「私」もいる。まあ，これ自体は極端に驚くようなことではない。目が覚めると，夢の中では疑問すら抱かなかった事柄が急に不思議に思えてきたりするものだ。そういえば，夢の中で見ていた数日前の夢は突然，静止したり，逆再生したりしてたっけ……。 (U)

1990年1月号12月18日(月)発売

特集　オペレーティングスタイルの研究
プログラムから環境変数を設定する/OS-9の使い方
第2特集　C言語プログラミング応用編
3Dタートルグラフィック/正規表現サーチの実際/GCC活用
1989年度GAME OF THE YEARノミネート
X1用シミュレーションゲーム　Super Battle
特別付録　X68000全ゲームカタログ

バックナンバー常備店

東京	神保町	三省堂神田本店5F 03(233)3312
	〃	書泉ブックマートBI 03(294)0011
	〃	書泉グランデ5F 03(295)0011
	秋葉原	T-ZONE 7Fブックゾーン 03(257)2660
	八重洲	八重洲ブックセンター3F 03(281)1811
	新宿	紀伊国屋書店本店 03(354)0131
	高田馬場	未来堂書店 03(200)9185
	渋谷	大盛堂書店 03(463)0511
	池袋	リブロ池袋店 03(981)0111
	〃	西武百貨店9F コンピュータ・フォーラム 03(981)0111
神奈川	横浜	有隣堂横浜駅西口店 045(311)6265
	〃	有隣堂ルミネ店 045(453)0811
	藤沢	有隣堂藤沢店 0466(26)1411
神奈川	厚木	有隣堂厚木店 0462(23)4111
	平塚	文教堂四の宮店 0463(54)2880
千葉	柏	新星堂カルチェ5 0471(64)8551
	船橋	リブロ船橋店 0474(25)0111
	〃	芳林堂書店津田沼店 0474(78)3737
	千葉	多田屋千葉セントラルプラザ店 0472(24)1333
埼玉	川越	黒田書店 0492(25)3138
	川口	岩渕書店 0482(52)2190
茨城	水戸	川又書店駅前店 0292(31)0102
大阪	北区	旭屋書店本店 06(313)1191
	都島区	駸々堂京橋店 06(353)2413
京都	中京区	オーム社書店 075(221)0280
愛知	名古屋	三省堂名古屋店 052(562)0077
	〃	パソコンΣ上前津店 052(251)8334
	刈谷	三洋堂書店刈谷店 0566(24)1134
長野	飯田	平安堂飯田店 0265(24)4545
北海道	室蘭	室蘭工業大学生協 0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は，とじ込みの振替用紙の「申込書」欄に何年何月号からをご記入のうえ，**年間購読料6,720円**(税込)を添えてお申し込みください。その際，裏面の通信欄に「○年○月号よりOh!X定期購読希望」と忘れずに明記してください。なお，すでに定期購読をご利用いただいている方には，購読期限終了と同時にご通知申し上げますので，同封の払込用紙をご利用ください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(238)0700

Oh!X 12月号
■1989年12月1日発行　定価560円(本体544円)
■発行人　孫　正義
■編集人　橋本五郎
■発売元　(株)日本ソフトバンク
■出版事業部　〒102 東京都千代田区九段南2-3-26　井関ビル
Oh!X編集部　☎03(230)7681
出版営業部　☎03(230)7670 FAX 03(262)8397
広告営業部　☎03(230)7672
■印　刷　凸版印刷株式会社

BACK ISSUES バックナンバー案内

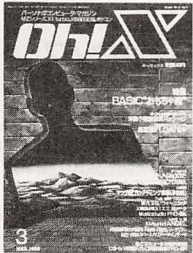
ここには1988年12月号から1989年11月号までをご紹介しました。現在1987年4，1988年1,2,4～9，1989年1～11月号までの在庫がございます。バックナンバーおよび定期購読のお申し込み方法については本文166ページを参照してください。

1988

12月号(品切れ)
特集　パソコンはいま音楽の領域へ
なぜ自動作曲か/心地よい雑音の話/和音の読み方/美しい響きの要素/4分音符は歌い始める/古くて新しい音楽形式/FM音源の仕組み/Melody Box/MusicBASIC
●さよなら LIVE in '88　バッハ イタリア組曲 他6本
●Oh!X　1周年記念特別企画「ちょっとあぶない福袋」
OS-9/X68000入門(2)　OS-9のオペレーション環境
Z80マシン語ゲーム工房/C調言語講座PRO-68K
全機種共通システム　ソースジェネレータSOURCERY

1989

1月号
特集　いきなり初春からハードウェア
デジタル回路入門/電子サイコロ/乱数発生器/X1turboバンクメモリ拡張/X68000用CP/M-80システム　他
1988年度GAME OF THE YEAR ノミネート作品発表
●MZ-2500用　Hyper Game Book
●LIVE in '89 エンデューロレーサー/アルルの女
●ようこそ，セガ・メガドライブ!!
C調言語講座PRO-68K/Z80マシン語ゲーム工房
全機種共通システム　パズルゲーム LAST ONE/FLICK

2月号
特集　マシン語"でじたるざんまい"
アーキテクチャからのマシン語入門/アセンブラへの招待/超入門Z80マシン語活用術/X68000料理教室
THE SOFTOUCH　彩CRONE/Final Ver.3.2　他
●X1/X1turbo用RPG FLAME
Z80マシン語ゲーム工房　最終回　爆発，そして完成へ
C調言語講座PRO-68K (8) とおりゃんせなのである
OS-9/X68000入門(3)　ついに発売！OS-9/X68000
全機種共通システム　高速エディタアセンブラREDA

3月号
特集　BASIC"おもちゃ箱"
ピコピコゲームから重力シミュレーションまで
●X1/X1turboでMZ-700用スペハリ/ロボットゲームTAMA
●数値演算を高速化　FLOAT2＋.X
OS-9/X68000入門(4)　C言語の概要を見る
C調言語講座PRO-68K(9)　ニホン語，不得意
新連載予告編X68000マシン語プログラミング入門
全機種共通システム 浮動小数点演算パッケージSOROBAN
THE SOFTOUCH/LIVE in'89/知能機械概論/猫とコンピュータ

4月号
特集　ゲーマーたちの"新深夜族"宣言
1988年度GAME OF THE YEAR
新連載　X68000マシン語プログラミング
●X1/X1turbo用パズルゲーム　ロボット衛兵
●MZ-700用ゲームパッケージ System-7B
●LIVE グラディウスII/ザ・スキーム/パワードリフト
連載　C調言語講座PRO-68K/OS-9/X68000入門
全機種共通システム　SLANG用実数演算ライブラリ
特別付録　X68000イメージCGポスター

5月号
特集　MIDIサウンドデータ料理術
LA音源をFM音源でシミュレート/X-BASICでMIDI制御
特別企画　第4回「言わせてくれなくちゃだワ」
●シャープパソコンフォーラム'89 in赤坂
●詳解Human68k ver.2.0
●MZ-2500，X1/X1turbo用　戦略的ライトサイクルゲーム
連載　C調言語講座PRO-68K/ OS-9/X68000入門
X68000マシン語プログラミング
全機種共通システム　ソースジェネレータRING

6月号
特集　これからのXfamily
X68000に光磁気ディスクを/学習リモコンの製作
THE SOFTOUCH　ライトニングバッカス/Might and MagicII他
●OPMA用外部関数による　KENBAN.BAS
●X1/X1turbo用ドライブゲーム　Spirit of Rally
●X1turboZ用　これ，パズルなんですか。
MZ-2500 MIDI入門(1)MIDIボードを作る
C調言語講座PRO-68K/X68000マシン語プログラミング
全機種共通システム　超小型コンパイラTTC

7月号
特集 3Dグラフィックへの飛翔
Zバッファアルゴリズム/スムースシェイディング 他
THE SOFTOUCH Terazzo PRO-68K/アドヴァンスト・ファンタジアン
新連載
DōGA・CGアニメーション講座
MZ-2500用グラフィックエディタ作成講座
マシン語カクテル in Z80's Bar
X-BASICプログラミング調理実習
全機種共通システム TTC用パズルゲームTIC BAN
X68000マシン語プログラミング/C調言語講座PRO-68K 他

8月号
特集1　X1プログラミングガイドブック
PCGの基礎から奥義まで/超高速ラインルーチン 他
特集2　3Dグラフィックの深淵へ
スキャンラインZバッファ/3Dモデリング 他
新連載 (で)のショートプロぱーてぃ
X68000マシン語プログラミング/C調言語講座 PRO-68K
X-BASICプログラミング調理実習/DōGA・CGA講座
MZ-2500用グラフィックエディタ/Z80's Bar 他
全機種共通システム CP/M用ファイルコンバータ

9月号
特集　活用ハードディスク&プリンタ
各社ハードディスク接続総チェック/ハードディスク雑学講座/COPYキーメニュー/ビデオプリンタ活用プログラム　他
THE SOFTOUCH　ジェノサイド/琉球/mFORTH Compiler
●サイバースティックで遊ぶ　不思議な環境ソフトの世界
●X1/X1turbo用シューティングゲーム　Defeat X
Z80's Bar/MZ-2500グラフィックエディタ　他
[X68000] X-BASIC/マシン語/C調言語講座/DōGA・CGA
全機種共通システム　生物進化シミュレーションBUGS

10月号
特集　ゲーム面白心理学
ソーサリアン・宇宙からの訪問者/ファンタジーゾーンねじ式/ガウディ・バルセロナの風/サバッシュ　他
●MZ-700用シューティングゲームSide Roll-F
●X1/X1turdo用カードゲームBonding
ショートプロ/Z80's Bar/MZ-2500グラフィックエディタ
X68000マシン語/X-BASIC/C調言語講座/DōGA・CGA
THE SOFTOUCH　Z'sTRIPHONY DIGITAL CRAFT/James68K
全機種共通システム　小型インタプリタ言語TTI

11月号
特集　microComputer入門
初歩からのCPU物語/RISCプロセッサの設計と製作
X68000&X1で周辺LSIを使いこなそう
連載
ショートプロ/Z80's Bar/MZ-2500グラフィックエディタ
X68000マシン語/X-BASIC/C調言語講座/DōGA・CGA
●X68000用カードゲームばばぬき
LIVE in '89　メタルホーク/オブ・ラ・ディ，オブ・ラ・ダ
THE SOFTOUCH　Stationery PRO-68K/リングマスター1
全機種共通システム　TTI用パズルゲームPUSH BON!

パソコン・AV専門 O.A.ランド

- お近くの方は、お立寄り下さい。専門係員がアドバイスいたします。
- ビジネスソフト、ゲームソフトのことならおまかせ下さい!!

セール期間
◀ '89 11・16▸12・15

涼しいときには、ハイ・パピプペポンと ウィンターセール 実施中!!

安心と信頼のOAランド・優良パソコン販売店、
アフターサービス万全のサポート体制。

NEW ランド特選 SHARP X68000EXPERT・EXPERT HDセット

X68000EXPERT HDセット 40MB HDD内蔵 2MB RAM

- CZ-612C……定価￥466,000
- CZ-612D……定価￥119,800
- MD-2HD 20枚サービス

クレジット例:12回…月々￥39,000、24回…月々￥20,400

他店には負けません!! 合計定価￥585,800

現金大特価!! (安いぞ)

ゲームソフト 5ゲームプレゼント

組合せは自由ダョ!!

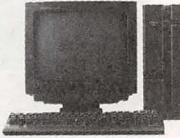
X68000EXPERTセット 2MB RAM内蔵

- CZ-602C……定価￥356,000
- CZ-612D……定価￥119,800
- MD-2HD 20枚サービス

クレジット例:12回…月々￥31,500、24回…月々￥16,500

OAランドで買わなきゃ損をする! 合計定価￥475,800

現金大特価!! (大推選!!)

ゲームソフト 5ゲームプレゼント

NEW X-1ターボZⅢセット CRTクリーナー キーボードカバープレゼント

Ⓐセット

- CZ-888CBK…定価￥169,800
- CZ-880DBK… 定価￥109,800
- CZ-6ST1-B…定価￥ 5,800 (チルトスタンド)
- MD-2HD 20枚サービス

合計定価￥275,400

現金価格 **特価中TEL下さい**

安すぎてゴメンなさい!

Ⓑセット

- CZ-888CBK…定価￥169,800
- CZ-830DBK…定価￥ 98,000
- CZ-6ST-1B…定価￥ 5,800 (チルトスタンド)
- MD-2HD 20枚サービス

合計価格￥273,600

合計価格 **特価中TEL下さい**

NEW SHARP X68000 PRO・PRO HDセット

ゲームソフト 5ゲームプレゼント

X68000PROセット

- CZ-652C……定価￥298,000
- CZ-612D……定価￥119,800
- MD-2HD 20枚サービス

クレジット例:12回…月々￥27,800、24回…月々￥14,500

合計定価￥417,800

現金特価!! TEL下さい。

X68000PRO-HDセット

- CZ-662C……定価￥408,000
- CZ-612D……定価￥119,800
- MD-2HD 20枚サービス

クレジット例:12回…月々￥34,900、24回…月々￥18,300

合計定価￥527,800

現金特価!! TEL下さい。

X1 周辺機器 お買徳!! ワープロ

①CZ-8DT2(デジタルテロッパー) 定価￥498,000…特価￥ 2,500
②CZ-81EBS(拡張I/Oボックス) 定価￥ 29,800…特価￥ 20,000
③CZ-500H(10MB HDD) 定価￥348,000…特価￥ 45,000
④CZ-53F(増設ドライブ) 定価￥ 19,800…特価￥ 9,000
⑤CZ-503F(ミニフロッピー) 定価￥ 49,800…特価￥ 29,800
⑥WD-901(カラーワープロ) 定価￥298,000…特価￥ 85,000
●VC-S500 (S-VHSビデオ) 定価￥145,000…特価￥ 78,000

①東芝JW-90B(ワープロ) 定価￥148,000…特価￥ 68,000
②エプソンLQX(ワープロ) 定価￥198,000…特価￥ 89,000
③OASYS FROM 11D(ワープロ) 定価￥138,000…特価￥ 49,800
④OASYS 30LX(ワープロ) 定価￥198,000…特価￥129,000

周辺機器コーナー

X1用

品番	定価	特価
CZ-8BV2	￥ 39,800	￥ 31,000
CZ-8BR1	￥ 29,800	￥ 23,000
CZ-8DT2	￥ 44,800	￥ 35,000
CZ-8BS1	￥ 23,800	TEL下さい
CZ-8TM2	￥ 49,800	￥ 38,000
CZ-8EB3	￥ 33,800	￥ 27,000

X68000用

品番	定価	特価
CZ-6PU1A	￥ 38,000	￥ 30,000
CZ-6BM1	￥ 26,800	￥ 21,000
CZ-6BE1	￥ 88,000	￥ 69,800
CZ-6VT1	￥ 69,800	TEL下さい
CZ-8NS1	￥188,000	￥149,000
CZ-6BC1	￥ 79,800	￥ 63,000

プリンターセットコーナー

品名	定価	特価
①CZ-6PU1(カラービデオプリンター)	￥198,000	￥152,000
②CZ-8PC3(カラープリンター)	￥ 65,800	￥ 53,000
③CZ-8PK8(ドットプリンター)	￥152,000	￥115,000
④CZ-8PK7(ドットプリンター)	￥122,000	￥ 93,000
⑤PC-PR201TH(カラープリンター)	￥145,000	￥103,000
⑥PC-PR201G(ドットプリンター)	￥158,000	￥ 99,000

その他、周辺機器・プリンター ソフトウェアー 20%〜25% OFF!!

X68000用ソフトウェアー・コーナー

品名	定価	特価
①CZ-212BS(BUSINESS)	￥ 68,000	￥ 53,000
②CZ-220BS(DATA)	￥ 58,000	￥ 45,000
③CZ-215MS(Sampling)	￥ 17,800	￥ 13,800
④CZ-221HS(NEW Print Shop)	￥ 10,800	￥ 15.500
⑤CZ-227BS(TOP財務会計)	￥200,000	￥158,000
⑥CZ-226BS(CARD)	￥229,800	￥ 23,000
⑦CZ-223CS(Communication)	￥ 19,800	￥115,500
⑧CZ-213MS(MUSIC)	￥ 18,800	￥ 14,800
⑨CZ-211LS(C compiler)	￥ 39,800	￥ 31,000
⑩C-TRACE(キャスト)	￥ 68,000	￥ 52,000
⑪EW(イースト)	￥ 38,000	￥ 29,000

■ハードディスク ■特価品もありますのでTEL下さい。

- アイテック IT-MJ4(I/F付)……特価￥98,000
- アイテック IT-MJ4 C(I/F付)……特価￥109,000
- ウィンテック HD-404HS(I/F付)…特価￥108,000
- コンピュータ CRC-MH4(I/F付)……特価￥70,000
- スナイパー SR-340Ⅱ(I/F付)……特価￥78,000
- アイテック ITH-320S(I/F付)……特価￥79,800
- ウィンテック HD-202(I/F付)……特価￥58,000
- スナイパー SR-520(I/F付)……特価￥55,000
- コンピュータ CRC-HD2A(I/F付)…特価￥62,000
- ロジテック LHD-32NR(I/F付)……特価￥80,000

今月の特価品 各一台限り その他、いろいろありますのでTEL下さい!!

■A紙品(美品・POP品) ■B級品(キズ少々) ■C級品(キズ有り)

シリーズ	品番	A級品	B級品	C級品
X68000シリーズ	CZ-612C	￥338,000	￥325,000	￥310,000
	CZ-652C	￥219,000より	￥212,000	￥203,000
	CZ-611D	￥ 90,000	￥ 86,000	￥ 80,000
	CZ-603	￥ 58,000	￥ 55,000	￥ ――
X-1シリーズ	CZ-888C	￥ 99,800より	￥ 90,000	――
	CZ-822C	￥ 24,000より	￥ 20,000	――
	CZ-880D	￥ 75,000	￥ 71,000	――
	CZ-830C	￥ 37,000	￥ 33,000	――
X-1プリンター	CZ-8PC3	￥ 48,000	￥ 45,000	￥ 42,000
	CZ-7PK7	￥ 83,000	￥ 75,000	￥ 45,000
	CZ-8PK8	￥ 98,000	￥ 85,000	￥ 62,000
	CZ-6PV1	￥138,000	￥134,000	￥125,000

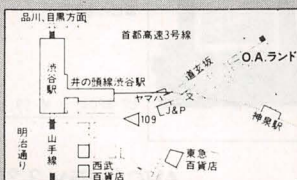
中古パソコン(価格・在庫は変動します。予約は5日以内といたします。)

機種	価格	機種	価格
PC-9801RA2	￥285,000より	CZ-652C	￥198,000より
PC-9801RA5	￥380,000より	CZ-612C	￥298,000より
PC-9801RX2	￥208,000より	CZ-888C	￥108,000より
PC-9801VX2	￥195,000より	CZ-880C	￥ 65,000より
PC-9801VM2	￥148,000より	CZ-500H	￥ 38,000より
PC-9801UV21	￥138,000より	CZ-620H	￥ 75,000より
PC-9801UV11	￥158,000より	PC-8801MA、H	￥ 79,000より
PC-9801VF2	￥ 85,000より	PC-8801FA、H	￥ 69,000より
PC-9801F2	￥ 68,000より	PC-8801SR	￥ 55,000より
PC-9801LT11	￥ 88,000より	FM77AV40	￥ 49,000より
PC-9801LV21	￥148,000より	FM77AV20EX	￥ 45,000より
PC-9801XL2	￥275,000より	PC-KD854	￥ 40,000より
PC-286V	￥148,000より	PC-KD853	￥ 47,000より
PC-286VE	￥158,000より	200ラインCRT	￥ 12,000より
PC-286L	￥138,000より	400ラインCRT	￥ 32,000より
PC-286LE	￥148,000より	400ラインTV付	￥ 45,000より
CZ-600C	￥158,000より	80桁プリンタ	￥ 25,000より
CZ-611C	￥205,000より	136桁プリンタ	￥ 38,000より

通信販売のご案内

全国通販

■銀行振込で申し込みの方は商品名及びお客様の住所・氏名・電話番号をお知らせ下さい。
〔振込先〕第一勧業銀行 渋谷支店
普通No.1163457 ㈱オーエーランド
■現金書留で送金されるお客様は電話番号と商品名、数量を明記して同封して下さい。■クレジットでご購入を希望される方は申し込み用紙をお送り致しますのでご記入の上返送して下さい。20才以上の方は、原則として保証人不要です。クレジットは1〜60回払で月々5,000円よりご自由に設定できます。

品川、目黒方面 / 首都高速3号線 / O.A.ランド / 渋谷駅 / 井の頭線渋谷駅 / 道玄坂 / ヤマハ / J&P / 109 / 神泉駅 / 明治通り / 山手線 / 西武百貨店 / 東急百貨店

- 下取・買取は電話で見積りしております。責任を持って下取りさせて頂きます。
- ご注文、お問合せは…毎日午前10時から午後7時まで
- 商品のお届けは…入金確認後、即日発送致します。

㈱オーエー・ランド

〒150東京都渋谷区円山町20-4 第5日新ビル1F
☎(03)770-8855 FAX (03)770-7080

関東エリアの送料は、1個につき￥1,000です。

★全商品保証書付。専門のアドバイザーが、お客様のニーズに対応します。
★初期不良・輸送トラブル等に迅速に対応し、即交換させていただきます。

■表示価格は、税別表示です。詳しくは、お電話にて、お問い合せ下さい。掲載の価格は、9月末現在です。

秋葉原価格でローンができます
電気の街秋葉原で 25年の信用!!

AVCフタバ

AUDIO VIDEO COMPUTER

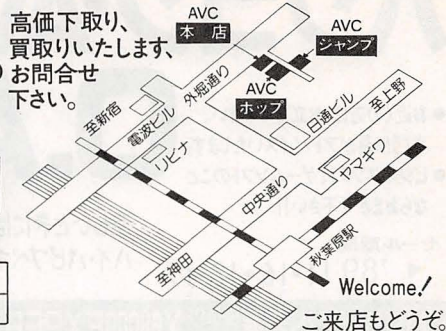
☎03(253)7661

〒101 東京都千代田区外神田3-2-3 ☎03-253-7611(代)

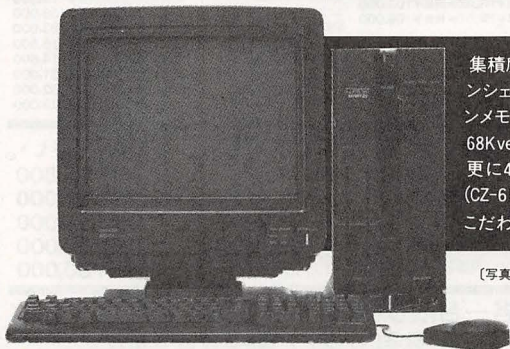
高価下取り、買取りいたします、お問合せ下さい。

AVC本店　AVCジャンプ　AVCホップ　至新宿　電波ビル　外堀通り　リバナ　日通ビル　至上野　ヤマギワ　中央通り　至神田　秋葉原駅

Welcome! ご来店もどうぞ。

今すぐ もよりの電話から	仙 台 022-264-3704	名古屋 052-452-3271	広 島 082-295-6873
札 幌 011-611-5104	新 潟 0252-75-4175	大 阪 06-311-3931	福 岡 092-481-2494

X68000の情報のすべて!(当店はX68000の認定代理店です。お気軽にご相談下さい)

X68000 待望の新しい仲間登場!!

X68000 PERSONAL WORKSTATION EXPERT・EXPERT HD

集積度を高めた"マンハッタンシェイプ"2Mバイトのメインメモリを標準実装、Human68Kver2.0搭載(CZ-602C)更に40MBのHDDを搭載(CZ-612C)あくまでもX68Kにこだわるマシン。

〔写真のモニタは別売です。〕

CZ-602C 標準価格¥356,000
CZ-612C 標準価格¥466,000
AVC特価

X68000 PERSONAL WORKSTATION PRO・PRO HD

拡張I/Oスロットを4スロット標準装備、メインメモリ1MB、Human68Kver2.0搭載(CZ-652C)更に40MBのHDDを搭載(CZ-662C)新しいX68Kの発見があるはずだ。

〔写真のモニタは別売です。〕

CZ-652C 標準価格¥298,000
CZ-662C 標準価格¥408,000
AVC特価

X68000 PERSONAL WORKSTATION ACE・ACE HD

グレーのみ、5台限り

従来機も忘れずに!!

CZ-611C-GY··¥399,800
CZ-603D-GY·¥ 84,800
合計··········¥484,600
AVC特価
➡¥279,800

お勧めディスプレイコーナー 組合せは自由、価格はお気軽にご相談下さい。

CZ-612D 標準価格¥118,800 AVC特価
●0.31mmドットピッチ ●TVチューナ搭載 ●3モードオートスキャン ●チルト台同梱

CZ-603D 標準価格¥84,800 AVC特価
●0.31mmドットピッチ ●TVチューナ無し ●3モードオートスキャン ●チルト台同梱

CZ-602D 標準価格¥99,800 AVC特価
●0.39mmドットピッチ ●TVチューナ搭載 ●3モードオートスキャン ●チルト台同梱

CU-21CD 標準価格¥139,800 AVC特価
●0.52mmドットピッチ ●TVチューナ無し ●3モードオートスキャン ●チルト台取付不可

型 番	品 名	標準価格	販売価格
CZ-6TU	システムチューナー	¥ 33,100	AVCフタバ特価
BF-68PRO	CRTフィルター	¥ 19,800	AVCフタバ特価
CZ-8NS1	カラースキャナー	¥188,000	AVCフタバ特価
CZ-6BN1	スキャナー用パラレルボード	¥ 29,800	AVCフタバ特価
CZ-6VT1	カラーイメージユニット	¥ 69,800	AVCフタバ特価
CZ-8BV2	カラーイメージボード	¥ 39,800	AVCフタバ特価
CZ-8BR1	立体映像セット	¥ 29,800	AVCフタバ特価
CZ-8DT2	パーソナルテロッパ	¥ 44,800	AVCフタバ特価
CZ-8BS1	FM音源ボード	¥ 23,800	AVCフタバ特価
CZ-8NJ1	ジョイカード	¥ 1,700	AVCフタバ特価
CZ-8NM2A	マウス	¥ 6,800	AVCフタバ特価
CZ-8NM3	マウス・トラックボール	¥ 9,800	AVCフタバ特価
CZ-6SD1	システムラック	¥ 44,800	AVCフタバ特価
AN-S100	アンプ内蔵スピーカー	¥ 36,600	AVCフタバ特価
CZ-6EB1	拡張I/Oボックス	¥ 88,000	AVCフタバ特価
CZ-8PC3	24ドットカラープリンター	¥ 65,800	AVCフタバ特価
CZ-8PK7	24ピンプリンタ(80桁)	¥122,000	AVCフタバ特価
CZ-8PK8	24ピンプリンタ(136桁)	¥152,000	AVCフタバ特価
CZ-8PK9	24ピンプリンタ(80桁)	¥ 89,800	AVCフタバ特価
IO-735X	カラージェットプリンター	¥248,000	AVCフタバ特価
AP-800	48ドットカラープリンター(エプソン)	¥ 99,800	¥?9,000
VP-1000	24ピン(136桁)(エプソン)	¥154,000	¥?8,000
AP-550	24ドットカラープリンター(エプソン)	¥ 69,800	¥?9,000
CZ-6BE1A	1MB増設RAMボード	¥ 38,000	AVCフタバ特価
CZ-6BE2	2MB増設RAMボード	¥ 79,800	AVCフタバ特価
CZ-6BE4	4MB増設RAMボード	¥138,000	AVCフタバ特価
CZ-6BP1	数値演算プロセッサボード	¥ 79,800	AVCフタバ特価
CZ-6BC1	FAXボード	¥ 79,800	AVCフタバ特価
CZ-6BM1	MIDIボード	¥ 26,800	AVCフタバ特価
CZ-6BU1	ユニバーサルI/Oボード	¥ 39,800	AVCフタバ特価
CZ-8TM2	モデムユニット	¥ 49,800	AVCフタバ特価
CZ-252MS	Musicstudio	¥ 28,800	AVCフタバ特価
CZ-247MS	MUSIC(MID)	¥ 28,800	AVCフタバ特価
CZ-221HS	NEW Print Shop	¥ 19,800	AVCフタバ特価
CZ-228BS	TOP給与計算エキスパート	¥200,000	AVCフタバ特価
CZ-227BS	TOP財務会計	¥200,000	AVCフタバ特価
CZ-220BS	DATA	¥ 58,000	AVCフタバ特価
CZ-212BS	BUSINESS	¥ 68,000	AVCフタバ特価
CZ-219SS	OS-9	¥ 29,800	AVCフタバ特価
CZ-211LS	Ccompiler	¥ 39,800	AVCフタバ特価
CZ-234LS	AI-68K	¥188,000	AVCフタバ特価
CZ-620H	20MBハードディスク	¥178,000	AVCフタバ特価
CZ-64H	40MBハードディスク	¥120,000	AVCフタバ特価
LHD-34V	40MBハードディスク(ロジテック)	¥153,000	¥117,000
LHD-32V	20MBハードディスク(ロジテック)	¥128,000	¥ 98,000

CZ-8NJ2

アナログジョイステック
標準価格 ¥23,800

AVC特価¥???

X1turboZIII

X1ターボシリーズの独自の機能を全継承。VCCIゼロdB基準に適合させた。

CZ-888C···¥169,800
CZ-860D···¥ 99,800
合計·······¥269,600
特価 ???

応談 価格はご相談に応じます。電話でお問い合せ下さい。

CZ-8PC4

48ドット熱転写プリンター。精密な文字、ハードコピーも可能。

CZ-8PC4·····¥ 99,800

AVC特価¥???

IT X640

40MBハードディスク、OS-9、Human68Kの使用可。

アイテック
IT X640·······¥158,000
特価 ¥118,000

特価¥118,000

●但し消費税(3%)は別途請求させていただきます。●分割回数は3回~48回まで自由に選べます。

●セットの組合せは自由!広告に出ていない他の機種はお問合せ下さい。

●頭金なし(手軽な電話クレジット)●製品先取り(お支払いは約1~2ヶ月後から)●低金利クレジット(1回の支払いは2,700円以上上で3~48回。ボーナス併用も可)●カレッジクレジット(保証人なし。但し満20歳以上の学生の方)●18歳未満の方(ご両親が代理購入者としてお申し込み下さい)●納期(通常の場合、当社に申込書が到着後1週間以内。特に人気のある商品で品薄の場合、少々納期が遅れることがありますので御了承下さい)●完全保証(すべてメーカー保証書付。アフターケア万全)●全国代引(お届けした者に、代金をお支払いいただく方法です。但し手数料1,000円)

AM10時からPM7時まで受付 日曜・祝日も営業

クリエイト特典

- ●全商品完全保証書付(メーカー保証)
- ●全国無料配達(一部離島の方は有料になります)
- ●配達日の指定OK(日曜・祭日にかかわらずお客様のご都合にあわせて配達します)
- ●どんな商品の組合せも自由自在(ご予算、用途に応じ自由自在にシステムアップできます)
- ●中古パソコン高額下取り(今お使いのパソコンをわずかな差額でグレードアップ)
- ●お支払い方法自由(低金利の均等払い、ボーナス一括払いもご利用ください)

営業時間(年中無休)
AM10:00～PM7:00(日曜・祭日はPM6:00まで)

当社はX68000の販売認定店です。どんなことでも安心してご相談ください。

(セットでお買い上げのお客様にお好きなゲームソフトとテレホンカードを差し上げます。)

夢のつづきを語ろう。

X68000 PRO

基本セット

- ●CZ-652C(本体・キーボード・マウス) ……… ¥298,000
- ●CZ-602D(カラー専用ディスプレイ) ……… ¥ 99,800
- ●CZ-8PC3(熱転写カラー漢字プリンタ) ……… ¥ 65,800
- ●プリンタ用紙・ブランクディスケット ……… ¥サービス
- ■定価合計 ……… ¥463,600

クリエイト特価

均等払い	¥ 5,480×36回	¥ 3,970×48回	¥ 3,460×60回
ボーナス	¥30,000× 6回	¥25,000× 8回	¥20,000×10回

X68000 EXPERT

格安基本セット

- ●CZ-602C(本体・キーボード・マウス) ……… ¥356,000
- ●CZ-603D(カラー専用ディスプレイ) ……… ¥ 84,800
- ●ブランクディスケット(5"2HD・10枚) ……… ¥サービス
- ■定価合計 ……… ¥440,800

クリエイト特価

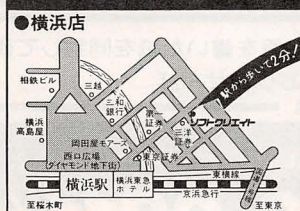
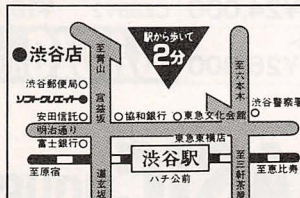
均等払い	¥ 6,190×36回	¥ 4,710×48回	¥ 4,210×60回
ボーナス	¥25,000× 6回	¥20,000× 8回	¥15,000×10回

※本広告に掲載の全商品の価格について消費税は含まれておりません。

冬のパソコン祭り

11月24日(金)～12月10日(日)

★期間中に限りクレジットの利息は無料です。(10回払いのみ)

①新品大謝恩特価セール!
②ソフト20%～30%OFF!
③特別高額下取りセール!
④サプライ用品大特売!
⑤周辺機器大特価セール!

X68000シリーズ用 周辺機器・ソフトお買い得セール

型番	品名	定価
CZ-6VT1	カラーイメージユニット	¥ 69,800
CZ-8NS1	カラーイメージスキャナ	¥188,000
CZ-6BE1A	IMB増設RAMボード	¥ 38,000
CZ-6BE2	2MB増設RAMボード	¥ 79,800
CZ-6BE4	4MB増設RAMボード	¥138,000
CZ-6BU1	ユニバーサルI/Oボード	¥ 39,800
CZ-6BG1	GP-IBボード	¥ 59,800
CZ-6BP1	数値演算プロセッサ・ボード	¥ 79,800
CZ-8NT1	トラックボール	¥ 13,800
CZ-6BM1	MIDIボード	¥ 26,800
CZ-6EB1	拡張I/Oボックス(4スロット)	¥ 88,000
CZ-8NJ2	アナログスティック	¥ 23,800
CZ-603D	ドットピッチ0.31mm14型高解像度	¥ 84,800
CZ-6TU	パソコンチューナ	¥ 33,100

ソフト名	品名	定価
MUSIC PRO-68K	マウスを使った楽譜ワープロ	¥ 18,800
SOUND PRO-68K	サウンドエディタ	¥ 15,800
Sampling PRO-68K	AD PCMサンプリングエディタ	¥ 17,800
Musicstudio PRO-68K V.1.1	MIDIマルチレコーディングソフト	¥ 28,800
NEW Print Shop PRO-68K	ポップアートツール	¥ 19,800
Communication PRO-68K	高機能通信ソフト	¥ 19,800
OS-9/X68000	マルチタスクオペレーティングシステム	¥ 29,800
AI-68K	AI開発ツール	¥188,000
BUSINESS PRO-68K	統合型計算ソフト	¥ 68,000
DATA PRO-68K	コマンド型リレーショナルデータベース	¥ 58,000
CARD PRO-68K	カード型リレーショナルデータベース	¥ 29,800
TOP財務会計	プロフェッショナル財務会計ソフトウェア	¥200,000
Ccompiler PRO-68K	ソフト開発セット	¥ 39,800
Human 68K Ver2.0	開発ツールセット	¥ 9,800

▲上記以外ビジネスソフト、最新ゲームソフト豊富に在庫あります。※送料はご注文の際お問合せください。●超特価販売中!

●渋谷店

●横浜店

総合お問合せ先☎03-486-6541(代)

パソコン専門ショップ

ソフトクリエイト 渋谷/横浜

●渋谷店☎03-486-6541(代) 〒150:東京都渋谷区渋谷1-12-7 三和渋谷ビル 振込銀行:三井銀行 渋谷宮益坂支店(普)No.5000340

●横浜店☎045-314-4777(代) 〒221:横浜市神奈川区鶴屋町2-12-8 第1建設ビル 振込銀行:三和銀行 横浜駅前支店(普)No.310852

おかけさまで9周年

9周年感謝フェア

もう2度と出せないかもしれない超特価！

AMIGA CALL

Macintosh CALL

X68000 EXPERT

CZ602C
CZ612D
Disk Cacher

標準特価 ¥481,600-を
BH特価 ¥385,000-で！

X68000 PRO

CZ652C
CZ603D
Disk Cacher

標準価格 ¥386,000-を
BH特価 ¥311,000-で！

X1 turboZ III

CZ888C
CZ860D
チルトスタンド

標準価格 ¥268,400-を
BH特価 ¥218,000-で！

ディスプレイ	プリンタ			スキャナ	
シャープ CZ602D ¥84,000	シャープ CZ8PC3 ¥56,000	シャープ CZ-8PK9 ¥74,000	エプソン VP135EX ¥79,000	シャープ CZ8NS1 ¥143,800	エプソン GT3000V ¥110,000
シャープ CZ612D ¥99,000	シャープ CZ8PC4 ¥86,800	シャープ CZ-6PV1 ¥172,000	エプソン VP1000 ¥108,000	エプソン GT4000 ¥158,000	オムロン HS10RII ¥41,000
シャープ CZ603D ¥73,800	シャープ CZ-8PK8 ¥125,000	エプソン AP-800 ¥74,800	エプソン VP2000 ¥124,800	エプソン GT1000 ¥64,000	オムロン HS7RII ¥33,000

ハードディスク		ジョイスティック	拡張ボード		
シャープ CZ620H ¥88,000	ロジテック LHD32V ¥99,000	シャープ Cyber Stick ¥19,800	1M増設メモリ CZ6BE1A ¥34,000	増設RS232C CZ6BF1 ¥44,800	コプロセッサボード CZ6BP1 ¥71,800
シャープ CZ64H ¥105,000	アイテック ITX640 ¥118,000	電波新聞社 XE1ST ¥4,400	2M増設メモリ CZ6BE2 ¥71,800	MIDIボード CZ6BM1 ¥24,000	X1用カラーイメージボード2 CZ8BV2 ¥19,800
ロジテック LHD34V ¥122,000	アイテック ITX680 ¥159,800	電波新聞社 XE1PRO ¥8,400	4M増設メモリ CZ6BE4 ¥124,000	スキャナボード CZ6BN1 ¥26,800	BASIC HOUSE

モデム	
アイワ PVA1200 ¥16,800	オムロン MD2400F ¥44,800
オムロン MD1200AIII ¥15,800	エプソン SR-240AT ¥40,000

11月23日〜26日はBASIC HOUSE！

通信販売希望の方は、現金書留で（商品代金＋送料¥1,000）×消費税1.03分を住所、氏名、希望商品名等を書いた紙を同封して御申し込み下さい。長期クレジットOK、支払い方法は御相談に応じます。（表示価格に消費税は含まれておりません。）

全国どこでも発送可　長期クレジットOK　送料全国均一¥1,000　宅配便にて即日配送

株式会社計測技研　本社営業部／マイコンショップ／通販部 〒321宇都宮市竹林町503－1 TEL0286-22-9811 FAX0286-25-3970

マイコンショップ BASIC HOUSE　お申し込み・お問い合せは ☎0286-22-9811(代)

安心と信頼の誌上ショッピング メディアショップ

お申込みは今すぐ電話かハガキで!!

株式会社 メディアショップ ハイランド　〒239 神奈川県横須賀市ハイランド3-9-6

電話でのお申込みは

お申し込みはフリーダイヤルで(料金無料)

0120-483290

お問合せは専用ダイヤルで

0468-483290

年中無休AM10時～PM10時

ハガキでのお申込みは

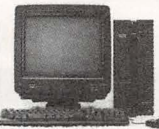
〒239
神奈川県横須賀市ハイランド3-9-6
(株)メディアショップ ハイランド
Oh!X 係

申込書
- 商品名(商品番号)
- 支払回数
- お名前
- 生年月日
- ご住所、電話番号
- お勤め先
  名称、住所、電話番号

通信販売のお申込み方法

▶現金一括でお申込みの方
- 商品名(商品番号)及び、住所、氏名、電話番号、ご覧の雑誌名をご記入の上、代金を現金書留でお送り下さい。
- 振込をご希望の方は、必ずお振込前にお電話又はおハガキで、お知らせ下さい。
  〈銀行振込〉協和銀行・久里浜支店　当座No.2945
  〈郵便振替〉横浜9-42177

▶クレジットでお申込みの方
- 電話かハガキでお申込み下さい。
  クレジット申し込み用紙をお送り致しますので、ご記入の上、当社へお送り下さい。

SHARP X68000 EXPERT

- CZ-602C(FDタイプ) 標準価格 356,000円
- CZ-612C(HDタイプ) 標準価格 466,000円
- CZ-602D(ディスプレイテレビ) 標準価格 99,800円
- CZ-612D(ディスプレイテレビ) 標準価格 119,800円
- CZ-603D(ディスプレイ) 標準価格 84,800円

SHARP X68000 PRO

- CZ-652C(FDタイプ) 標準価格 298,000円
- CZ-662C(HDタイプ) 標準価格 408,000円
- CZ-602D(ディスプレイテレビ) 標準価格 99,800円
- CZ-612D(ディスプレイテレビ) 標準価格 119,800円
- CZ-603D(ディスプレイ) 標準価格 84,800円

X68000 超特価セール!!

セットの組合わせも自由自在です。
まずはお問合わせ下さい。

EXPERT グラフィックス

- CZ-612C(本体)……466,000円
- CZ-612D(ディスプレイテレビ)……119,800円
- CZ-8NS1(イメージスキャナー)……188,000円
- CZ-6BN1(パラレルボード)……29,800円
- CZ-8PC4(48ドットカラープリンタ)……99,800円
- A-400HP(ビデオデッキ)……104,800円
- CZ-221HS(NEW Print SHOP)……19,800円
- CZ-235GS(グラフィックライブラリV.1)…8,800円
- CZ-236GS(グラフィックライブラリV.2)…8,800円

標準価格 1,045,600円

商品番号 227	一括払価格 814,000円
初回15,348円・12,500円×47回	ボーナス50,000円×8回
初回12,860円・10,800円×59回	ボーナス40,000円×10回

EXPERT サウンド【MIDI】

- CZ-602C(本体)……356,000円
- CZ-602D(ディスプレイテレビ)……99,800円
- AN-S100 (アンプ内蔵スピーカーシステム) 36,600円
- CZ-6BM1(MIDIボード)……26,800円
- MT-32(MIDI音源モジュール)……64,000円
- CZ-252MS(Musicstudio V1.1)……28,800円
- CZ-248MS(ソングライブラリ)……8,800円
- CZ-247MS(MUSICPRO68K MIDI)…28,800円

標準価格 649,600円

商品番号 228	一括払価格 542,000円
初回 9,444円・8,900円×47回	ボーナス30,000円×8回
初回 9,480円・8,300円×59回	ボーナス20,000円×10回

EXPERT 通信・パソコンFAX

- CZ-612C(本体)……466,000円
- CZ-603D(ディスプレイ)……84,800円
- BF-68PRO(CRTフィルター)……19,800円
- CZ-8TM2(モデムユニット)……49,800円
- CZ-8PK8(24ピン漢字プリンタ)……152,000円
- CZ-6BC1(FAXボード)……79,800円
- CZ-223CS(Communication)……19,800円

標準価格 872,000円

商品番号 219	一括払価格 694,000円
初回13,308円・11,100円×47回	ボーナス40,000円×8回
初回11,160円・9,900円×59回	ボーナス30,000円×10回

PRO ワープロ

- CZ-652C(本体)……298,000円
- CZ-603D(ディスプレイ)……84,800円
- BF-68PRO(CRTフィルター)……19,800円
- CZ-8PK8(24ピン漢字プリンタ)……152,000円
- EW(日本語ワープロソフト)……38,000円

標準価格 592,600円

商品番号 221	一括払価格 477,000円
初回 9,264円・7,200円×47回	ボーナス30,000円×8回
初回 8,230円・6,900円×59回	ボーナス20,000円×10回

PRO データベース

- CZ-662C(本体)……408,000円
- CZ-612D(ディスプレイテレビ)……119,800円
- CZ-8PC4(48ドットカラープリンタ)……99,800円
- CZ-8NS1(イメージスキャナー)……188,000円
- CZ-6BN1(パラレルボード)……29,800円
- CZ-220BS(DATA PRO68K)……58,000円
- CZ-226BS(CARD PRO68K)……29,800円

標準価格 933,200円

商品番号 229	一括払価格 757,000円
初回13,324円・11,900円×47回	ボーナス45,000円×8回
初回12,930円・10,400円×59回	ボーナス35,000円×10回

X1, X1 turbo, X68000シリーズ用周辺機器

カラービデオプリンタ

●CZ-6PV1
パソコンやビデオ機器に対応。64階調(485×480ドット)で再現する、昇華性染料熱転写方式を採用。

標準価格 198,000円

商品番号 149	一括払価格 163,000円
24回	初回 8,720円・7,700円×23回
36回	初回 5,862円・5,300円×35回

カラー イメージ スキャナー

●CZ-8NS1
高速、高精度でハイレベルな画像入力を実現。最大A4サイズの原稿をフルカラー読み取り可能。

標準価格 188,000円

商品番号 188	一括払価格 155,000円
24回	初回 8,800円・7,300円×23回
36回	初回 6,970円・5,000円×35回

48ドット熱転写カラー漢字プリンタ

●CZ-8PC4
精緻で略字のない高品位印字。美文書もアートワークも鮮やかに、美しさの48ドットカラープリンタ

標準価格 99,800円

商品番号 216	一括払価格 82,000円
12回	初回 7,440円・7,300円×11回
24回	初回 6,080円・3,800円×23回

カラー イメージジェット

●IO-735X
はがきからOHPフィルム、B4横サイズ対応。鮮明カラープリンタ。バッファメモリ(128KB)搭載。
●IO-73CX
信号ケーブル。

標準価格 253,500円

商品番号 232	一括払価格 210,000円
36回	初回 8,540円・6,800円×35回
48回	初回 9,620円・5,300円×47回

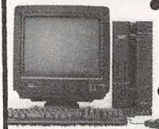
商品名	型式	標準価格	販売価格
14型カラーディスプレイ	CZ-603D	84,800	71,900
RGBシステムチューナー	CZ-6TU	33,1υ0	29,300
CRTフィルター	BF-68PRO	19,800	16,300
熱転写カラープリンタ	CZ-8PC3	65,800	54,000
漢字プリンタ(80桁)	CZ-8PK9	89,800	72,700
漢字プリンタ(80桁)	CZ-8PK7	122,000	98,400
漢字プリンタ(136桁)	CZ-8PK8	152,000	122,800
ハードディスク(20MB)	CZ-620H	178,000	143,700
増設用HDD(40MB)	CZ-64H	120,000	100,200
モデムユニット	CZ-8TM2	49,800	42,100

商品名	型式	標準価格	販売価格
カラーイメージユニット	CZ-6VT1	69,800	59,000
スキャナ用パラレルボード	CZ-6BN1	29,800	25,200
1MB増設RAM	CZ-6BE1	35,000	29,500
1MB増設RAM	CZ-6BE1A	38,000	32,100
2MB増設RAM	CZ-6BE2	79,800	67,300
4MB増設RAM	CZ-6BE4	138,000	116,400
ユニバーサルI/Oボード	CZ-6BU1	39,800	33,600
GP-IBボード	CZ-6BG1	59,800	50,400
増設用RS-232Cボード	CZ-6BF1	49,800	42,000
数値演算ボード	CZ-6BP1	79,800	67,300

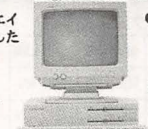
商品名	型式	標準価格	販売価格
FAXボード	CZ-6BC1	79,800	67,300
MIDIボード	CZ-6BM1	26,800	23,200
拡張I/Oボックス	CZ-6EB1	88,000	74,200
システムラック	CZ-6SD1	44,800	37,800
スピーカーシステム	AN-S100	36,600	30,900
カラーイメージボードII	CZ-8BV2	39,800	33,500
立体映像セット	CZ-8BR1	29,800	24,600
インテリジェントコントローラ	CZ-8NJ2	44,800	20,100
FM音源ボード	CZ-8BS1	23,800	20,100
フロッピーディスクユニット	CZ-503F	49,800	39,200

商品名	型式	標準価格	販売価格
DATA PRO68K	CZ-220BS	58,000	49,300
CARD PRO68K	CZ-226BS	29,800	25,400
Sampling PRO68K	CZ-215MS	17,800	15,300
NEW Print SHOP	CZ-221HS	19,800	16,400
Communication	CZ-223CS	19,800	16,900
C compiler	CZ-211LS	39,800	34,500
Musicstudio V1.1	CZ-252MS	28,800	24,600
MUSIC(MIDI)	CZ-247MS	28,800	24,600
OS-9/X68000	CZ-219SS	29,800	25,400
Stationery	CZ-240BS	14,800	13,700

今月の特選お買得品(限定)

SHARP X68000 EXPERT

●CZ-602C
ノーブルな機能美が、クリエイターを魅了する。妥協を排したこだわりのX68000。FDモデル。
●CZ-602D
15型カラーディスプレイテレビ
標準価格 455,800円

商品番号 223	一括払価格 359,000円
48回	初回9,888円・9,200円×47回
60回	初回8,810円・7,700円×59回

SHARP X68000 PRO

●CZ-652C
プロスペックと汎用性を絶妙にバランスさせたX68000。FDモデル。
●CZ-603D
14型カラーディスプレイ
標準価格 382,800円

商品番号 212	一括払価格 304,000円
48回	初回7,928円・7,800円×47回
60回	初回8,660円・6,500円×59回

安心と信頼 メディアショップ ハイランド

①完全保証 全国どこでもアフターケアOK
②全国無料配送 日曜配送可能
③支払回数は 予算に応じ3～60回 ボーナス併用可
④消費税 広告は全て消費税込みの価格で表示してあります
⑤FAXでも注文OK FAX：0468(48)3273
⑥その他広告以外の商品も取扱っております。お気軽にお問合せ下さい。

SHARP X68000 EXEショップ

歳暮謝恩セット大特価セール!

AIBIT
アイビット電子株式会社

X68000 PRO
PERSONAL WORKSTATION

本体+ディスプレイ

CZ-652C(本体) 定価¥298,000
CZ-603D(ディスプレイ) 定価¥ 84,800
定価合計¥382,000

セット大特価¥300,000
(税・送料込)

●代金は商品引換着払いでお願いします。

上記の他、X68000格安で販売中!

シャープMZ-1X30 モデムホン
(1X19上位機種)

標準価格¥98,000⇨
特価¥39,800

〈300/1200BPS全2重通信対応モデム内蔵〉
〈音声入出力端子付〉
〈ダイヤルパルス/プッシュボタン対応〉
〈プッシュボタン音解析機能〉
〈シャープ手順、CCITT、V25bis通信手順サポート〉

ハガキもOK、NewMZプリンタ
漢字カラー熱転写プリンタ
シャープMZ-1P22

標準価格¥59,800⇨
特価¥38,640(ケーブル付)

〈24×24ドット漢字・7色カラー・漢字30字/秒高速印字・MZ1P17とフルコンパチ・5KBのバッハメモリ付〉
適応パソコン→MZ2000、2500、5500、6500シリーズ、X1シリーズ、X68000シリーズ他。

"プリンタ・コピー・ファクス" 1台3役のスグレモノ
パソコンファクス **MZ-1V01**

●MZ25セット(インターフェース ソフト付)
標準価格合計¥342.800⇨ **¥168,000**
●PC98セット(インターフェース ソフト付)
標準価格合計¥377.800⇨ **¥198,000**
●MZ-1V01本体のみ
標準価格 ¥278.000⇨ **¥120,000**
※上記セットをご注文の際は3.5か5インチのご指定をしてください

東芝J3100SS+クレオBUSI COMPOセット

J3100SS 定価¥198,000
BUSI COMPO 定価¥ 40,000
定価合計¥238,000

セット大特価¥195,000

アイビット推奨ディスプレイ

●富士通ゼネラルDM405
(14型)
(2000アナログ21/8ピン)
定価¥67,800⇨
特価¥36,000

DM405対応パソコン機種:MSX2。X1シリーズ。MZ700/1500/2000/2200シリーズ。FM77AV/7/8シリーズ。(ケーブルは各専用のものを使用)

●シャープCZ-830D・BK
(14型)
2モードオートスキャン方式
(アナログ/デジタル)
定価¥98,000⇨
特価¥54,800

CZ-830D対応パソコン機種:CZ880C/881C。X1/TURBOシリーズ。ケーブルは本体付属を使用。NEC PC-8801・9801シリーズ(XA・XLのみ不可)MZ700/1500/2000/2200/2500各シリーズ(推奨品シャープ8D8K)。

●シャープCZ-611D-GY
(15型アナログTV/3モードオートスキャン)
¥145,000→**¥89,800**

CZ-611D対応パソコン機種:※X1シリーズ/※X1 turboシリーズ/X1 yurboZシリーズ/X68000シリーズ/PC8801シリーズ/PC-9801シリーズ/PC-286シリーズ
(※は接続ケーブルAN1506が必要です)

●三菱XC-1498C
(14型アナログ/ドットピッチ0.28mm)
定価¥99,800⇨
特価¥54,800

XC-1498C対応パソコン機種:
NEC・PC9801シリーズ。
エプソンPC286/386シリーズ。

本 体/新旧在庫機種(新品)
●シャープ/CZ-602C/CZ-612C/CZ-652C/CZ-662C/CZ-801C/CZ-802C/CZ-803C/CZ-804C/CZ-820C/CZ-822C/CZ-888C/MZ-2200/MZ-2861/MZ-3500/MZ-5511/MZ-6556
●富士通/FM-NEW7/FM77AV/FM77AV1/FM77AV2/FM77AV20/FM77AV40/FM77D2/FM77L2/TOWNS1/TOWNS2
●東芝/J-3100SL/J-3100SS
●NEC/PC9801CV21/PC9801E/PC9801LV21/PC9801RA2/PC9801RX2/PC9801UV21/PC9801VX4/PC9801XA2

拡張機器他
●シャープCZ-8GR(X1.GRAM)‥¥32,000⇨¥12,000
●シャープCZ-52F(F増設ドライブ代品)………¥15,000
●シャープCZ-51F(ターボ増設ドライブ代品)…¥15,000
●シャープCZ-8EP(I/Oポート)…・¥11,800⇨¥9,000
●シャープCZ-8EB3(I/Oボックス)‥¥33,800⇨¥28,000
●シャープCZ-8BK3…・(X1)…・¥13,800⇨¥11,700
●シャープCZ-8BK4…・(X1)……・¥6,800⇨¥5,700
●シャープCZ-8BGR2・(X1)…・・¥14,800⇨¥4,000
●シャープCZ-8BS1…・(X1)…・¥23,800⇨¥19,500
●シャープCZ-64H(ハードディスク)〈CZ-602・652内蔵用〉…¥120,000
●シャープCZ-8NJ2(インテリジェントコントローラー)¥23,800⇨大特価
●シャープCZ-8SS2システムスタンド…¥5,500⇨¥2,500
●シャープCZ-81Tチルトスタンド……・¥8,500⇨¥1,000
●シャープCZ-8RL1(データーレコーダー)…¥24,800⇨¥16,000
●シャープMZ-1U08(1500,700拡張)‥¥25,000⇨¥12,000
●シャープMZ-1U03(1500,700拡張)‥¥35,000⇨¥15,000
●シャープMZ-1X22モデムユニット…¥21,800⇨¥13,000
●シャープMZ-1R12 RAM………¥35,000⇨¥8,000
●シャープMZ-1E29 (MZ)……・¥17,800⇨¥9,800
●シャープMZ-1U09…(2500)…・¥9,000⇨¥7,200
●シャープMZ-1M03…(5500)・¥69,000⇨¥35,000
●シャープMZ8BC04‥(2000)‥¥18,000⇨¥8,000
●シャープMZ-8B104‥(2000)・¥45,000⇨¥18,000
●シャープMZ-1R11…・(5500)・¥80,000⇨¥40,000
●シャープMZ-1R24…(2500)‥¥22,000⇨¥6,000
●シャープMZ-1R26A‥(2500)・¥13,000⇨¥12,800
●シャープMZ-1R27A‥(2500)・¥13,000⇨¥10,000
●シャープMZ-1R28A・(2500) ¥13,000⇨¥10,000
●シャープMZ-1R29A‥(2500)・¥32,000⇨¥10,000
●シャープMZ-1T02…(2000)‥¥19,800⇨¥8,500
●シャープMZ-1T03…(1500)‥¥12,000⇨¥8,500
●シャープMZ-1X29…………¥13,800⇨¥11,000
●シャープMZ1R35(1MB増設メモリーボード)‥¥55,000⇨¥19,000
●シャープMZ1R36(1R35用増設1MBボード)¥45,000⇨¥15,000
●シャープMZ1E26(ボイスコミュニケーション)…¥24,800⇨¥13,000
●シャープMZ-3500キーボード……………¥10,000
●シャープMZ-5500キーボード……………¥10,000
●シャープ2000/2200キーボード…………¥10,000
●シャープSS-SC28M(ハンディーコピーキット)¥49,800⇨¥10,000
●シャープ1E35(ADPCMボード)……・¥49,800⇨¥13,000
●シャープ1E39(RE232C 2CHボード)…・¥39,800⇨¥13,000
●シャープX1、MZ用マウス……………特価¥4,800
●シャープX1用ジョイカード…………………¥1,500
●富士通16βキーボード……・¥25,000⇨¥20,000

プリンター
●シャープCZ-8PK7(漢字プリンター)・¥122,000⇨¥97,600
●シャープCZ-8PK8(漢字プリンター)¥152,000⇨¥121,500
●シャープCZ-8PK9(漢字プリンター)…¥89,800⇨¥71,800
●シャープCZ-81P(801用プロッタプリンタ)……・¥3,500
●シャープCZ-8PC2…………¥69,800⇨¥46,800
●シャープCZ-8PC3…………¥65,800⇨¥52,000
●シャープCZ-8PC4(黒・グレー)・¥99,800⇨大特価
●シャープCZ-8PD3(X1用)……¥59,800⇨¥16,000
●シャープMZ-1P27………¥268,000⇨¥214,400
●シャープMZ-1P28………¥148,000⇨¥118,400
●シャープMZ-1P29………¥168,000⇨¥134,400
●シャープ6P-11(カットシードヒート)…・¥95,000⇨¥35,000
●富士通FMPR-201…………¥79,800⇨¥45,000
●富士通FMPR-351………¥250,000⇨¥125,100
●富士通FMPR-353………¥198,000⇨¥115,000
●富士通MB-27413…………¥90,000⇨¥25,000
●富士通FMPR-201R1(二水ROM)¥23,000⇨¥11,000
●富士通MB27407(熱転写プリンタ)…¥79,800⇨¥33,000
●NEC-NM9700(漢字プリンタ)‥¥163,000⇨¥88,000

ディスプレー(カラー)
●富士通FMTV-211(200)…・¥185,000⇨¥89,000
●富士通FMTV-152(200)…・¥109,000⇨¥58,000
●NEC PC-KD854(400)……・¥89,800⇨¥58,000

ディスプレー(モノカラー)
●シャープMZ-1D10(400)…・¥41,800⇨¥25,000

フロッピーディスク
●シャープCZ-503F…………¥49,800⇨¥34,000
●シャープCZ-503F(インターフェースカードなし)…・¥30,000
●シャープCZ-502F…………¥99,800⇨¥75,000
●シャープCZ-300F(CZ-3PCM付)…………¥13,000

ソフト
●春望クリエイティブII・(2500)・¥34,800⇨¥29,000
●ビジレス……………(2500)・¥48,000⇨¥42,000
●Hu-CAL日本語……(2500)・¥45,000⇨¥30,000
●ぷりんとしょっぷ……(2500)…¥9,800⇨¥5,000
●FILE UTILITY UT-25F・(2500)…・¥6,800⇨¥6,000
●パーソナルCP/M6Z001(2500)……………11月入荷
●V2BASIC6Z010……(2500)‥¥10,000⇨¥8,500
●FORTRAN(1P1213)‥(2500)・¥13,800⇨¥11,700
●C MZ2500 1P1214‥(2500)・¥13,800⇨¥11,700
●COBOL 1P1215…・(2500)・¥13,800⇨¥11,700
●C CZ116LF(X1)……………¥13,800⇨¥11,700
●COBOL CZ118LF‥(X1)…・¥13,800⇨¥11,700
●ランゲージマスターCZ128SF…¥9,800⇨¥8,500
●シャープCZ-130F(ターボCP/M)…・¥14,800⇨¥12,500
●シャープCZ-133SF(モデムターミナル)‥¥25,000⇨¥6,500
●シャープX1・3インチCP/M…・¥16,800⇨¥5,000
●HUMAN68K CZ-244SS……・¥9,800⇨¥8,500
●シャープ1P-1251(MZ2861用)…¥88,000⇨¥20,000
●シャープ1P-1252(MZ2861用)‥¥55,000⇨¥20,000
●シャープ1P-1253(MZ2861用)‥¥77,000⇨¥20,000
●シャープ1P-1254(MZ2861用)‥¥66,000⇨¥20,000

X68000関係ソフト
●マイクロソフトウェアージャパン「C&プロフェッショナルパッケージ」…………… ¥58,000⇨¥49,800
●シャープOS-9/X68000……・¥29,800⇨¥25,300
●シャープCZ-211LS…………¥39,800⇨¥33,800
●シャープCZ-6BE1…………¥35,000⇨¥29,000
●シャープCZ-6BE1A…………¥38,000⇨¥32,000

シャープNEWパーソナルワークステーション
●PC/ATコンパチPC8041…………………¥998,000
●AXディスクトップMZ8302A………………¥278,000
●AXディスクトップMZ8306A………………¥458,000

《全商品新品完全保証付》 ■シャープポケコン全商品販売中。カタログ、特価表ご請求ください(〒72)。

☎0426-45-3001~3
FAX.0426-44-6002
●営業時間/10:00~19:00●電話受付/20:00迄可●定休日/日曜日(祭日営業)
SHARP SUPER XEX SHOP
アイビット電子株式会社 〒192 東京都八王子市北野町560-5

上記の広告商品はすべて店頭販売もしております。

全国通信販売
北海道から沖縄まで

★送料はご注文の際にお問い合わせ下さい。
★掲載の商品は、すべて新品、保証書付きです。
★掲載の商品は充分用意してありますが、ご注文の際は、在庫の確認の上、現金書留または、銀行振込でお申し込み下さい。全商品クレジットでも扱っております。
★お申し込みの際は必ず電話番号を明記して下さい。
★商品、品切れの節はご容赦下さい。

富士銀行八王子支店 (普)1752505

●本誌発売時には、上記価格よりさらにお求めやすい価格に変更されている場合があります。 ●上記商品価格には消費税は含まれておりません。全ての商品に対し、別途3%の消費税金がかかりますのでご了承ください。

冬もX68000の季節

X68000 ACE HD

CZ-611C-GYあとわずか！
20MBのハードディスクを搭載してPROの定価より安く！

在庫僅少売切れ御免。

こたつ・木枯し・68K

X68000シリーズ

EXPERT	定価~~¥356,000~~
EXPERT HD	定価~~¥466,000~~
PRO	定価~~¥298,000~~
PRO HD	定価~~¥408,000~~

各シリーズとも特価販売中！
T・ZONE 2Fにて。

ADO・TOYOMURA T・ZONE ティー・ゾーン

Micom Zone

2F 〒101 東京都千代田区外神田4-4-1 ☎257-2650

海外でも使える
「T・ZONE CLUB」
カード会員募集中!!
「オリエント」「UC」「マスター」カードが1つになった。
「ボーナス一括払い」OK！「通信販売」もお手軽にご利用頂けます。そのほか、便利でお得な特典がいっぱい！ 今がチャンス!!
詳しくは、店頭にてどうぞ!!

X68000をトータルサポート
T・ZONE 2F

SHARP Authorized………

X68000 PRO SHOP

電子手帳コーナー

電子手帳とパソコンなど

ステーショナリー
PRO68K
¥14,800
注：電子手帳との接続には別売ケーブルCE200L（¥2,500）が必要です。

増設OK CZ-620H

SHARP純正20MBHD

効能 ①HD内蔵タイプのX68000に増設可。
②すでにHDを接続していても増設可。
（シャープ以外のハードディスクの場合でもご相談下さい。）
③もちろん最初の1台としても安心。
④なかなかサマになるデザインです。

限定！
定価¥178,000 ⇒ *Special Price!*

OS-9/68000 for X68000

- □OS9/68000 (SHARP) ¥29,800
- □C＆PRO PACK (マイクロウェア) ¥58,000
- □Src Dbg (マイクロウェア) ¥39,800
- □MW-BASIC (マイクロウェア) ¥60,000
- □BTree09 (ARK) ¥36,000
  MW-BASIC用のISAM用B-Treeパッケージです。応用例として住所録と販売管理プログラムが付属。全ソースコード付です。（このソフトを動かすためにはMW-BASICが必要です。）
- □UD-CACHE (ARK) ¥16,000
  すべてのRBFデバイスに対応するキャッシュです。
- □FBU (ARK) ¥38,000
  ハード・ディスクバックアップユーティリティーです。巨大ファイルを分割バックアップしたり、日付管理を行なったバックアップもOK。
- □VSED (FORKS) ¥28,000
  OS9/68000で唯一オートバッファリングをサポートしたスクリーンエディタです。
- □CSG IMS……は今対応中です。もう少々お待ち下さい。

番外 Oh! FMコーナー

FMシリーズ用OS9に新ソフト登場！

日本ソフトバンク
DB-09 (FM-7, 77, AV, 11) ¥18,252！
OS 9上で走るリレーショナルデータベースマネージャーです。問い合わせ形式で取扱い簡単。
なんとCによる全ソース付。

IMAGE and TEXT'S Inc.
Plot it！ (FM-11) ¥16,500
OS 9上で走るプリント基板パターン設計用CAD。
なんとVTerm 25にも対応。

星光電子感謝セール ～12月末まで
日頃のご愛顧にお応えして一部商品を除き（OS9News、定価1万円以上の商品）All 20%OFFでご提供します。

T・ZONE正社員・長期アルバイト募集中！
☆お問い合わせは総務課鈴木まで（TEL 03-257-2630）

T・ZONE
8F オフィス
7F 工学書輸入
6F 開発器 衛星放送
5F HAM トランシーバー
4F トレードゾーン
3F IBMゾーン
2F パソコンソフトウェア
1F パーソナルOA
B1 Macintosh!

営業時間：AM10:30～PM7:00

下記T・ZONE各店でも扱っています。

宇都宮店：☎0286(63)4949	川口店：☎0482(68)7826	ラジオショップ：☎03(257)2643	横浜店：☎045(641)7741
大宮店：☎048(652)1831	東ラジ店：☎03(257)2694	パーツショップ：☎03(257)2655	静岡店：☎0542(83)1331

●マイコン通販利用の方へ：現金書留で送金される際は、住所、氏名、TEL番号、希望商品名（詳しく）を明記して下さい 振込を御希望の方は下記銀行へお願いします
尚、いずれも予めTELにて、御予約・送料確認の上御送金下さい （振込口座 埼玉銀行 秋葉原支店 当座2705 ㈱亜土電子工業）

☆この広告の提示価格には、消費税は含まれておりません。

資料請求券
oh!X
12月号

君の味方だ！パソコン通信

フリーランスは
通信業態。

僕はフリーランスのカメラマン。
J&P HOT LINEで
シャッターチャンスを見つけ出す。

ロケ地探し/BBS

バッチリ狙いの風景を探すなら、地元の人に聞くのがいちばん。

フリーランスのカメラマンはフットワークが問題だ。遠くへ足をのばすにしても、狙い通りの風景をどれだけ的確に見つけられるかが、写真の仕上がりを左右する。こういう時は、J&P HOT LINEのBBS。全国の仲間にピタリのポイントを聞き出せば、効率的というものなのだ。

撮影計画/データベース

ホテルの所在も現地のまつりも、J&P HOT LINEで下調べ。

さて、撮影旅行に出かける前に、J&P HOT LINEのホテル情報で、撮影地の宿のあたりをつけておこう。全国各地の有名ホテルが一覧できるから、便利この上なし。現地で時間があまった時のために、当日行われそうな地元の祭りも調べておける。うーん、ゆとりだね。

ポラ送り/X-MODEM

とりあえずの仕上がりをクライアントに見てもらう。再撮影はまた明日。

撮影は順調に進んだ。今日の写真のいくつかは、クライアントに気に入ってもらえそうだ。さっそく、試しに撮ったポラロイドをパソコンのデータに変換し、得意先のCUG内へX-MODEMで送信する。画像データもおくれるなんて、なんて便利なネットなんだろ。

打ち合わせ/電子メール

次は雑誌のグラビアページ ロケ先からメールで打ち合わせ。

フリーランスは、時間の活かし方が重要になる。次の仕事は、都内でモデルの撮影だ。編集部のEさんに、スタイリストについて連絡をする。いつも忙しいEさんには、好きな時間に見てもらえる電子メールは、なにより確実な連絡方法。ムダのないのがありがたい。

批評会/SIG

時にはアマチュア時代を思い出す。新鮮な感覚がよみがえる。

さて、仕事の時間が終わったら、趣味の世界へ没頭しよう。SIGは、J&P HOT LINEの中のもうひとつのネットワーク。テーマを絞った議論についつい夢中。仕事で使う車についてモーターネットでひとくさり弁舌をぶってしまった。ああ4WDが欲しい。

情報収集/OLT（チャット機能）

仕事のネタも雑談の中から。脈絡のなさがいいのです。

SIGに書きこみをしていたら、たまたま同時にアマチュアカメラマンのSくんがアクセスしていた。ちょうどいいやと彼をさそってOLT（オンライン・トーク）へ入りこむ。日本中の人たちと同時に複数でおしゃべりしあえるOLTは、なんといっても、パソコン通信のダイゴミですな。

J&P HOT LINEは全国90ヵ所のアクセスポイント。
2万人の仲間が、あなたの仲間になってくれます。

●パソコン/ワープロ通信ネットワークサービス

J&P HOT LINE

アクセスポイントは全国に90ヵ所。日本全国を網羅する、本格的な通信ネットワークです。

ご入会はスタータキットで
買ったその日からアクセスできます。

■申込書
〒556 大阪市浪速区日本橋5-6-7 上新電機株式会社
J&P HOT LINE事務局宛　TEL.(06)632-2521

■利用料金について
入会金/3,000円（スタータキット購入の代金から充当されます）
接続料/3分あたり20円（アクセスポイントまでの電話代は含みません）
※消費税3%が加算されます。

スタータキット申込書

お名前	
お電話番号	
ご住所	〒

お申込品　スタータキット（ソフトなし）
3,000＋90（消費税3%）＝¥3,090

スタータキットのお求めは、J&P各店でどうぞ。

店名	住所	電話
渋谷店	東京都渋谷区道玄坂2丁目28番4号	☎(03) 496-4141
町田店	東京都町田市森野1丁目39番16号	☎(0427)23-1313
八王子店	東京都八王子市旭町1番1号八王子そごう7F	☎(0426)26-4141
立川店	東京都立川市幸町4－39－1	☎(0425)36-4141
富山店	富山市双代町1番地	☎(0764)42-2131
金沢店	金沢市入江2－63	☎(0762)91-1130
大須店	名古屋市中区大須4丁目2－48	☎(052)262-1141
テクノランド	大阪市浪速区日本橋5丁目6番7号	☎(06) 634-1211
メディアランド	大阪市浪速区日本橋5丁目8番26号	☎(06) 634-1511
コスモランド	大阪市浪速区難波中2丁目1番17号	☎(06) 634-3111
ワープロランド	大阪市浪速区日本橋4丁目9番15号	☎(06) 634-1411
ビジネスランド	大阪市北区梅田1-1-3大阪駅前第3ビルB2	☎(06) 348-1881
阪急三番街店	大阪市北区芝田1-1-3阪急三番街B1	☎(06) 374-3311
高槻店	高槻市高槻町11番16号	☎(0726)85-1212
くずは店	枚方市楠葉花園町15番2号	☎(0720)56-8181
千里中央店	豊中市新千里東町1-3-204千里サンタウン3F	☎(06) 834-4141
摂津富田店	高槻市大畑町24－10	☎(0726)93-7521
寝屋川店	寝屋川市緑町4－20	☎(0720)34-1166
藤井寺店	藤井寺市岡2丁目1番33号	☎(0729)38-2111
岸和田店	岸和田市土生町2451－3	☎(0724)37-1021
さんのみや1ばん館	神戸市中央区八幡通3-2-16	☎(078)231-2111
西宮店	兵庫県西宮市河原町5－11	☎(0798)71-1171
姫路店	姫路市東延末1丁目1番住友生命姫路南ビル1F	☎(0792)22-1221
京都寺町店	京都市下京区寺町通仏光寺下ル恵美須之町549	☎(075)341-3571
京都近鉄店	京都市下京区烏丸通七条下ル東塩小路町702	☎(075)341-5769
和歌山店	和歌山市元寺町4丁目4番地	☎(0734)28-1441
奈良1ばん館	奈良市三条町478－1	☎(0742)27-1111
郡山インター店	大和郡山市横田693－1	☎(07435)9-2221

T4910217912569　雑誌02179-12